Licence académique Informatique

Théorie des Langages

http://lguezouli.univbatna.com/
ou
http://fac-sciences.univ-batna.dz/cs/enseignants/guezouli_larbi_site/index.html

Présenté par Dr Larbi GUEZOULI

Théorie des Langages

- Chapitre I: Les langages [2 séances]
  - Introduction et rappels mathématiques
  - Opérations sur les langages
  - Représentation des langages : grammaires et automates
  - Hiérarchie de Chomsky
- Chapitre II: Les langages réguliers [2 séances]
  - Propriétés des langages réguliers
  - Expression régulières
  - Passage des expression régulières aux automates et réciproquement
- Chapitre III: Les automates d'états finis [2 séances]
  - Automates déterministes et minimisation
  - Automates non déterministes et passage a un automate déterministe
- Chapitre IV: Les langages algébriques [2 séances]
  - Propriétés des langages algébriques
  - Les automates à pile

Théorie des Langages

- Chapitre V: Les langages à contexte lié [2 séances]
  - Définition et propriétés
  - Les automates à bornes linéaires
- Chapitre VI: Les machines de Turing [2 séances]
  - Notion de machine de Turing
  - Langages de type 0 et machine de Turing
  - Introduction à la calculabilité

Mode d'évaluation :
Examen final : coef 2
Contrôle Continu : coef 1

Chapitre I
Les Langages

Plan

1. Introduction et rappels mathématiques
2. Opérations sur les langages
3. Représentation des langages : grammaires et automates
4. Hiérarchie de Chomsky

CHAPITRE I
LES LANGAGES

1. Introduction et rappels mathématiques

- Un **langage** est un ensemble de phrases.
- Une **phrase** est une suite de mots.
- Un **mot** est une suite de symboles.
- Pour construire une phrase, on suit des règles appliquées aux mots. L'ensemble de ces règles forme une **grammaire**.

Un **mot** est une chaîne de symboles.

Définition

Un mot sur l'alphabet A est une relation:

$u: \{1,\dots,m\} \rightarrow A$

$i \mapsto u(i)$

tel que u est le mot, m est la taille du mot |u|, u(i) est le $i^{ème}$ symbole du mot.

Notons l'ensemble des mots sur un alphabet A^*

Le mot vide ε est commun entre tous les langages $|\varepsilon| = 0$ et $\varepsilon \in A^*$

Par convention: $A \subseteq A^*$

Adjonction d'une occurrence à droite d'un mot est définie de l'ensemble $A^*xA \rightarrow A^*$ tel que la relation qui définie le mot résultant est:

$ux: \{1,\dots,m+1\} \rightarrow A$

$i \mapsto ux(i)$

On peut écrire donc:

$ux(i) = u(i)$ pour tout $i \in \{1,\dots,m\}$

$ux(m+1) = x$

$|ux| = |u|+1$

La concaténation des mots est une opération définie de A^*xA^* vers A^*

$u.v: A^*xA^* \rightarrow A^*$

$(u,v) \mapsto u.v$

Par récurrence: $\forall u \in A^*$ et $\forall v \in A^*$

$u.\varepsilon = u$

$u.(vx) = (u.v)x$ pour tout $x \in A$

Exp.

ab.abba = (ab.abb)a
= ((ab.ab)b)a
= (((ab.a)b)b)a
= ((((ab.ε)a)b)b)a
= ((((ab)a)b)b)a
= ababba

Pour simplifier l'écriture on écrit:

uv au lieu de u.v

uvw au lieu de (u.v).w ou u.(v.w)

2. Opérations sur les langages

Un **langage L** sur un alphabet A est une partie de A^*: $L \subseteq A^*$

L'ensemble des langages sur A est noté **$P(A^*)$**

Exp.

$\varnothing \subseteq A^*$, $A \subseteq A^*$, $A^* \subseteq A^*$ sont des langages de A

$\{u\} \subseteq A^*$ est un langage de A, tel que $u \in A^*$

$\{\varepsilon\} \subseteq A^*$ est un langage de A, puisque $\varepsilon \in A^*$ (le mot vide).

2. Opérations sur les langages (suite)

Remarque: $A \subseteq A^* \subseteq P(A^*)$

Attention, on ne peut pas écrire **u** $\in$ **v** avec $u \in A^*$ et $v \in A^*$.

2.1. Somme des langages

Soient $L \subseteq A^*$ et $M \subseteq A^*$ alors

$L+M \subseteq A^*$ est un langage appelé aussi la **réunion** de L et M.

2.1. Somme des les langages (suite)

La somme est définie aussi:

$\forall u \in A^*$: $u \in L+M$ ssi $u \in L$ ou $u \in M$.

Propriétés de la somme:

Soient $L \subseteq A^*$, $M \subseteq A^*$ et $N \subseteq A^*$:

$L+\varnothing = L$ et $\varnothing+L = L$	(Neutralité)
$(L+M)+N = L+(M+N)$	(Associativité)
$L+M = M+L$	(Commutativité)
$L+L = L$	(Idempotence)
$M \subseteq N \Rightarrow L+M \subseteq L+N$	(Croissance)

2.1. Somme des les langages (suite)

Attention:

$L+M = L+N$ est vraie si $M = N$ ou bien

$M \subseteq L$ et $N \subseteq L$.

Une dérivé de l'opération "somme" est la **différence**:

$u \in L-M$ ssi $u \in L$ et $u \notin M$

2.1. Somme des les langages (suite)

Attention:

$(L-M)+M$ $= L$ si $M \subseteq L$

$= L+M$ sinon.

$(L+M)-M$ $= L$ si $L \cap M = \varnothing$

$= L-M$ sinon.

2.2. Somme généralisée de langages

Soit un **ensemble de langages** sur l'alphabet A indexé par un ensemble d'entiers naturels I.

Cet ensemble de langages est appelé aussi **suite** de langages.

La **réunion** des langages de cet ensemble est:

$$\sum_{i\in I} L_i \subseteq A^*$$

2.2. Somme généralisée de langages (suite)

La somme généralisée de l'ensemble de langages est définie somme suit:

$$\forall u \in A^* : u \in \sum_{i\in I} L_i \text{ ssi } \exists i \in I \text{ tel que } u \in L_i$$

2.2. Somme généralisée de langages (suite)

Propriétés de la somme généralisée:

$$\sum_{i\in\phi} L_i = \phi$$

$$\forall M \subseteq A^* : \sum_{i\in I} L_i \subseteq M \text{ ssi } \forall i \in I, L_i \subseteq M$$

$$\forall i \in I, L_i \subseteq M_i \Rightarrow \sum_{i\in I} L_i \subseteq \sum_{i\in I} M_i$$

$$\sum_{i\in I} (L_i + M_i) = \sum_{i\in I} L_i + \sum_{i\in I} M_i$$

$$\sum_{k\in I+J} L_k = \sum_{i\in I} L_i + \sum_{j\in J} L_j \text{ tel que I + J est la réunion de I et J}$$

2.3. Concaténation des langages

La **concaténation** de 2 langages L et M est un langage noté L.M $\subseteq$ A*. Chaque élément de ce langage est la concaténation d'un élément de L et un élément de M dans l'ordre.

$$\forall u \in A^* : u \in L.M \text{ ssi } \exists v \in L \text{ et } \exists w \in M$$

$$\text{tel que } u = v.w$$

2.3. Concaténation des langages (suite)

Propriétés de la concaténation des langages:

Soient $L \subseteq A^*$, $M \subseteq A^*$ et $N \subseteq A^*$:

$L.\varepsilon = L$ et $\varepsilon.L = L$	(Neutralité)
$(L.M).N = L.(M.N)$	(Associativité)
$M \subseteq N \Rightarrow L.M \subseteq L.N$ et $M.L \subseteq N.L$	(Croissance)
$L.\varnothing = \varnothing$ et $\varnothing.L = \varnothing$	(Nullité)
$L.(M+N) = L.M + L.N$ et	
$(M+N).L = M.L + N.L$	(Distributivité)

2.4. Itération des langages

Le **langage itéré** L* de L (appelé aussi Fermeture de Kleene de L) est défini comme suit:

Soit $L \subseteq A^*$

$L^* \subseteq A^*$: $L^* = \sum_{i \geq 0} L^i$

avec L^i est définit par récurrence:

$L^0 = \varepsilon$

$L^{i+1} = L^i .L$

2.4. Itération des langages (suite)

Propriétés de la puissance:

$\forall L \subseteq A^*$:

$L^1 = L$

$L^i L^j = L^{i+j}$

$L^i L = L L^i = L^{i+1}$

2.4. Itération des langages (suite)

Propriétés de l'itération:

$\forall L \subseteq A^*, \forall M \subseteq A^*$ et $\forall N \subseteq A^*$:

si $M \subseteq L^*$ et $N \subseteq L^*$ alors $M.N \subseteq L^*$ (Stabilité par concaténation)

si $M \subseteq L^*$ alors $M^* \subseteq L^*$ (Stabilité par itération)

si $M \subseteq L$ alors $M^* \subseteq L^*$ (Croissance)

$\phi^* = \phi$ et $\varepsilon^* = \varepsilon$

2.4. Itération des langages (suite)

Propriétés de l'itération (suite):

$$L^*L = LL^*$$

$$L^*L^* = L^{**} = L^*$$

$$L^* = \varepsilon + L^*L$$

$$(L+M)^* = (L^* + M^*)^* = (L^*M^*)^*$$

$$= L^*(LM^*)^* = (L^*M)^*L^*$$

3. Représentation des langages : grammaires et automates

3.1. Grammaires

Une **grammaire** permet de générer un langage.

Elle permet de définir l'ensemble des mots d'un langage en appliquant un certain nombre de règles.

3.1. Grammaires (suite)

Une **grammaire** est un quadruplet:

$G = \langle V_T, V_N, S, R \rangle$

tel que: V_T: Vocabulaire terminal;

V_N : Vocabulaire non-terminal;

Notons: $V = V_T \cup V_N$ appelé Vocabulaire.

R: Ensemble de règles de la grammaire appelées aussi règles de réécriture;

S: Axiome ou symbole de départ. $S \in V_N$

3.1. Grammaires (suite)

Le format d'une règle:

$u1 \rightarrow u2$

avec $u1 \in V^+$ et $u2 \in V^*$

Notations:

•Une règle $u1 \rightarrow u$ tel que $u \in V_T^*$ est appelée règle terminale.

•Si plusieurs règles ont la même partie gauche, on peut factoriser leurs parties droites en utilisant '|'

3.1. Grammaires (suite)

Exp:

$S \rightarrow ab$, $S \rightarrow aSb$, $S \rightarrow c$

On peut l'écrire comme suit:

$S \rightarrow ab|aSb|c$

3.2. Dérivation en une étape

Soit une grammaire $G = \langle V_T, V_N, S, R \rangle$ et deux éléments des vocabulaires $u \in V^+$ et $v \in V^*$

G permet de dériver v de u en une étape ($u \Rightarrow v$) ssi:

$u = xu'y$ (x, y peuvent être vides)

$v = xv'y$ (x, v', y peuvent être vides)

$u' \rightarrow v'$ est une règle de R

3.3. Dérivation en plusieurs étapes

v peut être dérivée de u en plusieurs étapes:

$v \overset{+}{\Rightarrow} u$: si v peut être dérivé de u par 1 ou plusieurs dérivation en une étape.

$v \overset{*}{\Rightarrow} u$: si v peut être dérivé de u par 0 ou plusieurs dérivation en une étape.

3.4. Langage généré par une grammaire

Un langage généré par une grammaire $G = \langle V_T, V_N, S, R \rangle$ est l'ensemble des mots sur V_T qui peuvent être dérivés à partir de S:

$L(G) = \{v \in V_T^* \mid S \overset{+}{\Rightarrow} v\}$

Remarque:

- Une grammaire définit un seul langage;
- Un langage peut être définit par plusieurs grammaires

Exemple 01

Soit la grammaire $G = \langle V_T, V_N, S, R \rangle$ tel que:

$V_T = \{a,b\}$, $V_N = \{S\}$, $R = \{S \rightarrow aS \mid bb\}$

$S \overset{*}{\Rightarrow} a^nS \Rightarrow a^nbb$ avec $n \in N$.

Donc:

le langage engendré par cette grammaire est

$L(G) = \{a^nbb \mid n \in N\}$

Exemple 02

Soit la grammaire $G = \langle V_T, V_N, S, R \rangle$ tel que:

$V_T = \{a,b\}$, $V_N = \{S\}$, $R = \{S \rightarrow aS \mid bS \mid a\}$

$S \overset{*}{\Rightarrow} uS \Rightarrow ua$ avec $u \in \{a,b\}^*$, $n \in N$ et $|u| = n$.

Donc:

le langage engendré par cette grammaire est

$L(G) = \{ua \mid u \in \{a,b\}^*\}$

Exemple 03

Soit la grammaire $G = \langle V_T, V_N, S, R \rangle$ tel que:

V_T = {un, une, le, la, étudiant, pomme, écrit, fille, cours}

$V_N = \{S, A, B, C, D, E, F, G\}$

$R = \{ S \rightarrow AB, A \rightarrow CE|DF, B \rightarrow GA, C \rightarrow$ une$|$la, $D \rightarrow$ un$|$le, $E \rightarrow$ pomme$|$fille, $F \rightarrow$ étudiant$|$cours, $G \rightarrow$ écrit$\}$

Exemple 03 (suite)

$S \Rightarrow AB \Rightarrow DFB \Rightarrow DFGA \Rightarrow DFGDF \Rightarrow$ un FGDF $\Rightarrow$ un étudiant GDF $\Rightarrow$ un étudiant écrit DF $\Rightarrow$ un étudiant écrit un F $\Rightarrow$ un étudiant écrit un cours

Cette phase appartient au langage L(G)

Exemple 04

Soit le langage $L = \{ab^n a \mid n \in N\}$

On cherche à construire une grammaire $G = \langle V_T, V_N, S, R \rangle$ qui génère ce langage.

$V_T = \{a,b\}$

$V_N = \{S, A\}$

$R = \{ S \rightarrow aAa, A \rightarrow \varepsilon | bA\}$

Exemple 05

Soit le langage $L = \{a^{2n}b^n \mid n \in N\}$

On cherche à construire une grammaire $G = \langle V_T, V_N, S, R \rangle$ qui génère ce langage.

$V_T = \{a,b\}$

$V_N = \{S\}$

$R = \{ S \rightarrow aaSb|\varepsilon\}$

3.5. Représentation des automates

Un automate est une machine qui accepte ou rejette une chaîne de caractères de A* en se basant sur son appartenance à un langage L.

Un automate est définit par:

- Un ensemble fini de symboles;
- Un ensemble fini d'états;
- Une fonction de transaction entre les états.

3.5. Représentation des automates (suite)

Un automate peut être représenté par:

- Diagramme de transitions;
- Table de transitions;
- Notation formelle.

3.5. Représentation des automates (suite)

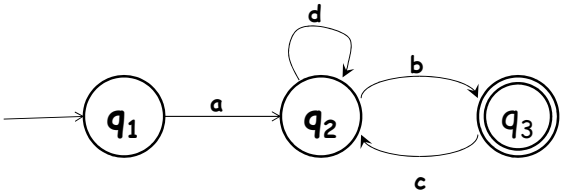
- **Diagramme de transitions**

C'est une représentation graphique sous forme d'un graphe orienté étiqueté.

Les nœuds: sont les états de l'automate (étiquetés par les noms des états, et l'état final représenté par double cercle);

3.5. Représentation des automates (suite)

Les arcs orientés: représentent les transitions de l'automates (étiquetés par les symboles, et l'état initial représenté par un arc sans nœud d'origine).



3.5. Représentation des automates (suite)

- **Table de transitions**

C'est une table contenant les états dans les lignes et les symboles dans les colonnes.

L'état initial représenté en ajoutant une →

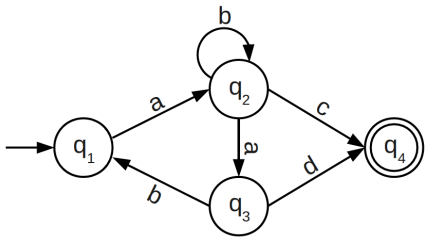
L'état final représenté en ajoutant une *

Une case contient l'état de destination de la transaction.

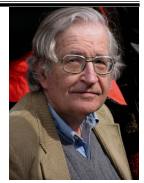
3.5. Représentation des automates (suite)

Exp.

	a	b	c	d
→q_1	q2	Ø	Ø	Ø
q2	q3	q2	q4	Ø
q3	Ø	q1	Ø	q4
*q4	Ø	Ø	Ø	Ø



4. Hiérarchie de Chomsky



En 1957, Chomsky Noam propose de classer les grammaire en 4 classes.

4.1. Grammaire générale (type 0)

Une grammaire est dite **générale** si toutes ses règles sont du format suivant:

$$u \rightarrow v, \text{ tel que } u, v \in V^*, u \neq \varepsilon$$

et u doit contenir au-moins un non-terminal

(Pas de restrictions sur les règles)

4.2. Grammaire sous-contexte (type 1)

Une grammaire est dite **sous-contexte** si toutes ses règles sont du format suivant:

$$uAv \rightarrow uBv, \text{ tel que } u, v \in V^*,$$

$$\text{et } A \in V_N, B \in V^+$$

$$\text{et } |uAv| \leq |uBv|$$

Le symbole non terminal A est remplacé par B si le contexte (u à gauche et v à droite) est présent.

4.3. Grammaire hors-contexte (type 2)

Une grammaire est dite **hors-contexte** si toutes ses règles sont du format suivant:

$$u \rightarrow v, \text{ tel que } u \in V_N, v \in V^*$$

Le membre gauche de chaque règle est un seul symbole non terminal.

4.3. Grammaire régulière à droite et à gauche (type 3)

Une grammaire est dite **régulière à droite** si toutes ses règles sont de l'un des deux formats suivants:

$$A \rightarrow uB, \text{ tel que } A,B \in V_N, u \in V_T$$

$$A \rightarrow u, \text{ tel que } A \in V_N, u \in V_T$$

La partie droite de chaque règle contient un symbole terminal, qui pourrait être suivi par un symbole non-terminal.

4.3. Grammaire régulière à droite et à gauche (type 3)

Une grammaire est dite **régulière à gauche** si toutes ses règles sont de l'un des deux formats suivants:

$A \rightarrow Bu$, tel que $A,B \in V_N$, $u \in V_T$

$A \rightarrow u$, tel que $A \in V_N$, $u \in V_T$

La partie droite de chaque règle contient un symbole terminal, qui pourrait être précédé par un symbole non-terminal.

4.4. Propriétés

Toute grammaire régulière est une grammaire hors-contexte.

Toute grammaire hors-contexte ne contenant pas d'ε-règle ($A \rightarrow \varepsilon$) est une grammaire sous-contexte.

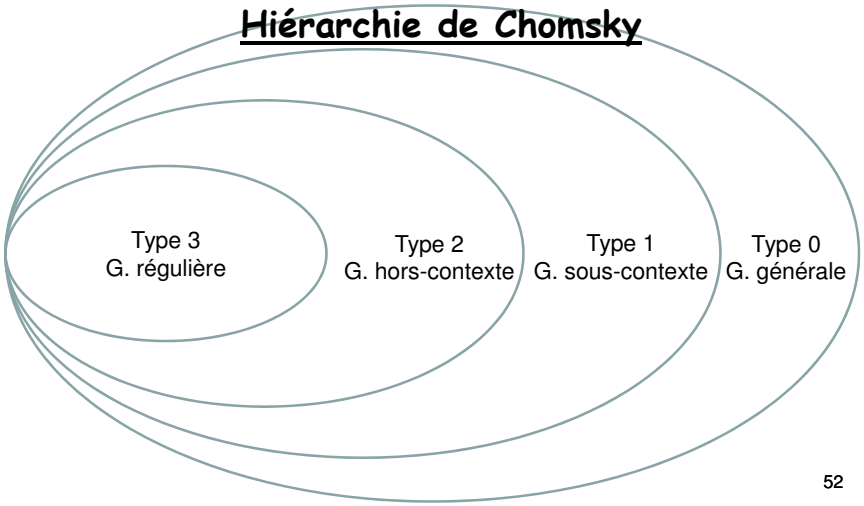
Tout langage hors-contexte ne contenant pas la chaîne vide ε est sous-contexte;

4.4. Propriétés

Un langage est dit **régulier à droite**, **hors-contexte** ou **sous-contexte** s'il existe une grammaire régulière à droite, hors-contexte ou sous-contexte qui l'engendre.

Un langage généré par une grammaire générale est dit **récursivement énumérable**.

Hiérarchie de Chomsky



CHAPITRE II
LES LANGAGES RÉGULIERS

PLAN

1. Propriétés des langages réguliers
2. Expression régulières
3. Passage des expression régulières aux automates et réciproquement
4. Grammaire et automates (grammaire de Kleene)

CHAPITRE II
LES LANGAGES RÉGULIERS

1. Propriétés des langages réguliers

Un langage est dit **régulier** ssi il existe une grammaire régulière qui génère ce langage.

On appel un langage régulier sur A tout langage définit de la façon suivante:

- $\varnothing$ est un langage régulier sur A;
- $\{\varepsilon\}$ est un langage régulier sur A;
- $\{a\}$ est un langage régulier sur A pour tout $a \in A$;

CHAPITRE II
LES LANGAGES RÉGULIERS

1. Propriétés des langages réguliers (suite)

- Si L1 et L2 sont des langages réguliers sur A, alors les langages suivants sont réguliers sur A:
  - $L1 \cup L2$
  - L1L2
  - $L1^*$ (fermeture de Kleen)
- Il n'y a pas d'autres règles qui définissent des langages réguliers.

CHAPITRE II
LES LANGAGES RÉGULIERS

1. Propriétés des langages réguliers (suite)

Exp.1

Pour tout $u \in A^*$, le langage $\{u\}$ est régulier.

Parce que:

$$u = u_1u_2...u_n$$

et le langage $\{u\}$ s'écrit :

$$\{u\} = \{u_1\}\{u_2\}...\{u_n\}$$

Comme les $u_i \in A$ alors les $\{u_i\}$ sont réguliers et leurs concaténations donne un langage régulier.

1. Propriétés des langages réguliers (suite)

Exp.2

Tout langage fini $L=\{u_1,u_2,\ldots,u_n\}$ est régulier, tel que $\forall i \in \{1,..,n\}$, $u_i \in A^*$.

Parce que le langage L s'écrit :

$$L = \{u_1\} \cup \{u_2\} \cup \ldots \cup \{u_n\}$$

Comme les langages $\{u_i\} \in A^*$ sont réguliers (Exp.1) alors leurs unions donne un langage L régulier.

1. Propriétés des langages réguliers (suite)

Exp.3

Le langage A^* est régulier.

Parce que:

$$A = \{u_1,u_2,\ldots,u_n\} = \{u_1\} \cup \{u_2\} \cup \ldots \cup \{u_n\}$$

est régulier, et A* est la fermeture de Kleen de A qui donne un langage régulier.

1. Propriétés des langages réguliers (suite)

Exp.4

Le langage $L=\{a^n b^m \mid n,m \in N\}$ sur l'alphabet {a,b} est régulier.

Parce que:

$$L = \{a^n \mid n \in N\} . \{b^m \mid m \in N\} = \{a\}^*\{b\}^*$$

L est la concaténation de deux fermetures de Kleen $\{a\}^*$ et $\{b\}^*$ qui sont des langages réguliers, donc L est régulier.

2. Expressions régulières

Une expression est une notation pour décrire les langages réguliers.

- $\varnothing$ est une expression régulière dénotant le langage régulier $\varnothing$;
- ε est une expression régulière dénotant le langage régulier $\{\varepsilon\}$;
- a est une expression régulière dénotant le langage régulier {a};

2. Expressions régulières (suite)

• Si E1 et E2 sont des expressions régulières dénotant les langages L1 et L2:

– E1+E2 est une expression régulière dénotant le langage régulier $L1 \cup L2$

– (E1E2) est une expression régulière dénotant le langage régulier L1L2

– $E1^*$ est une expression régulière dénotant le langage régulier $L1^*$

• Il n'y a pas d'autres expressions régulières

2. Expressions régulières (suite)

Notons:

• L(E) Le langage dénoté par l'expression régulière E.

• E1 et E2 équivalentes:

$E1 \equiv E2$ ssi L(E1) = L(E2)

Exp.1

E = (0+(1(0)*)) L(E) = {0,1,10,100,…}

2. Expressions régulières (suite)

Priorités:

• priorité (*) > priorité (.) > priorité (+)

• Pour tout langage régulier, on peut trouver une infinité d'expressions régulières le dénotant.

2. Expressions régulières (suite)

Exp.

L'expression $0+10^*$ est $(0+(1(0)^*))$

0^*10^* = {m ∈ {0,1}* | m contient un seul 1}

$(0+1)^*1(0+1)^*$ = {m ∈ {0,1}* | m contient au moins un 1}

$(0+1)^*001(0+1)^*$ = {m ∈ {0,1}* | m contient 001}

$((0+1)(0+1))^*$ = {m ∈ {0,1}* | |m| est pair}

2. Expressions régulières (suite)

Règles d'équivalence:

$a + (b + c) \equiv (a + b) + c$

$a + b \equiv b + a$

$a + \varnothing \equiv a$

$a + a \equiv a$

$a(bc) \equiv (ab)c$

$\varepsilon a \equiv a\varepsilon \equiv a$

$\varnothing a \equiv a\varnothing \equiv \varnothing$

2. Expressions régulières (suite)

Règles d'équivalence: (suite)

$a(b+c) \equiv ab + ac$

$(a+b)c \equiv ac + bc$

$\varepsilon + aa^* \equiv a^*$

$\varepsilon + a^*a \equiv a^*$

$\varnothing^* \equiv \varepsilon$

2. Expressions régulières (suite)

Exp.

$\varepsilon + 0 + 00 \equiv (\varepsilon + 0)(\varepsilon + 0)$

Parce que:

$\varepsilon + 0 + 00 \equiv \varepsilon + 0 + 0 + 00$ $(a + a \equiv a)$

$\equiv \varepsilon\varepsilon + \varepsilon 0 + 0\varepsilon + 00$ $(\varepsilon a \equiv a$ et $a\varepsilon \equiv a)$

$\equiv \varepsilon(\varepsilon + 0) + 0(\varepsilon + 0)$ $(a(b+c) \equiv ab+ac)$

$\equiv (\varepsilon + 0)(\varepsilon + 0)$ $((a+b)c \equiv ac + bc)$

2. Expressions régulières (suite)

Quelques équivalences utiles:

$(ab)^*a \equiv a(ba)^*$

$(a^*b)^*a^* \equiv (a + b)^*$

$a^*(ba^*)^* \equiv (a + b)^*$

$(\varepsilon + a)^* \equiv a^*$

$aa^* \equiv a^*a$

3. Expressions régulières et automates

Théorèmes:

- Pour tout automate, il existe une expression régulière reconnaissant le même langage.
- Pour toute expression régulière, il existe un automate reconnaissant le même langage.

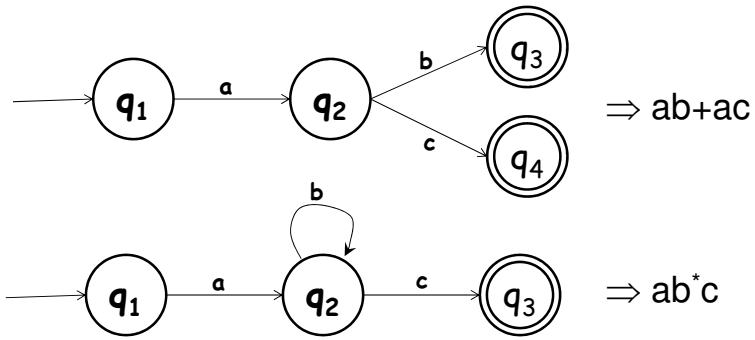
3. Expressions régulières et automates (suite)

Passage Automate → Expression régulière:

- Remplacer ',' par '+';
- Elimination des états intermédiaires (sauf initiaux et finaux);

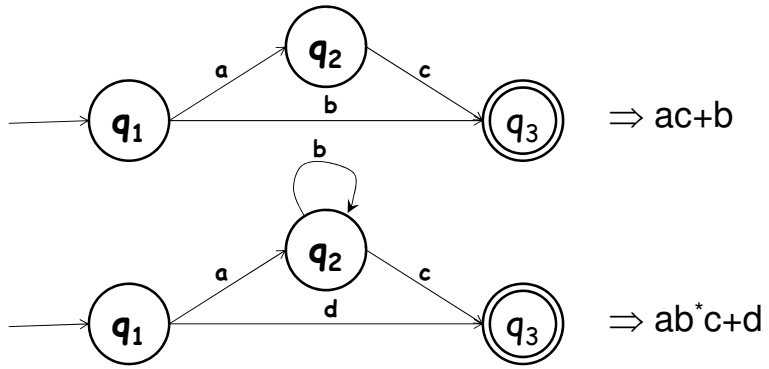
3. Expressions régulières et automates (suite)

Exp.



3. Expressions régulières et automates (suite)

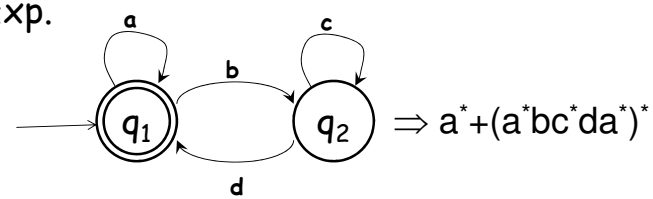
Exp.



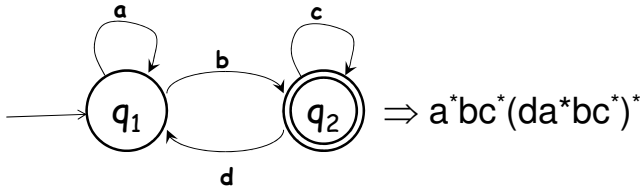
CHAPITRE II
LES LANGAGES RÉGULIERS

3. Expressions régulières et automates (suite)

Exp.



$\Rightarrow a^*+(a^*bc^*da^*)^*$



$\Rightarrow a^*bc^*(da^*bc^*)^*$

CHAPITRE II
LES LANGAGES RÉGULIERS

3. Expressions régulières et automates (suite)

Passage Expression régulière → Automate :

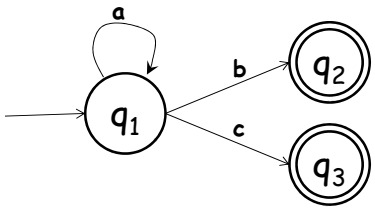
- Séparer les parties de l'expression pour créer les nœuds intermédiaires de l'automate ;

CHAPITRE II
LES LANGAGES RÉGULIERS

3. Expressions régulières et automates (suite)

Exp. Soit l'expression régulière suivante:

$a^*(b+c)$



CHAPITRE III
LES AUTOMATES D'ÉTATS FINIS

PLAN

1. Automates déterministes et minimisation
2. Automates non déterministes et passage a un automate déterministe

1. Automates déterministes et minimisation

1.1. Automates finis

Un **automate fini** (d'états finis) est une machine (**algorithme**) qui fait des calculs en utilisant une **mémoire de taille limitée**. Dans notre cas, ces calculs sont la reconnaissance de langages (déterminer si un mot appartient à un langage).

Ce type d'automate permet de reconnaitre les langages réguliers.

1.1. Automates finis (suite)

Exp.

Soit un automate fini reconnaissant les chaînes de caractères.

Une chaîne commence et termine par ' " '

Toute occurrence de ' " ' à l'intérieur d'une chaîne est doublée.

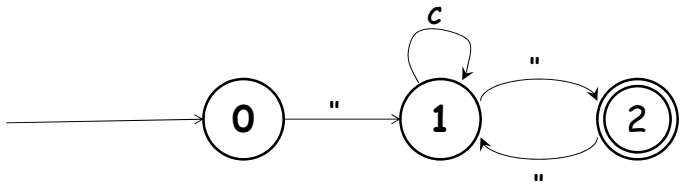
$S \rightarrow$ "(*C*'"')*"

$C \rightarrow$ a,b,c,…

1.1. Automates finis (suite)

Exp. (suite)

Sous forme graphique:



1.1. Automates finis (suite)

Définition formelle

Un automate A fini est un quintuplet $A = \langle Q, V, \delta, I, F \rangle$ tel que:

Q : est un ensemble fini d'états;

V : l'ensemble de symboles (vocabulaire);

$I \subseteq Q$: l'ensemble des états initiaux;

$F \subseteq Q$: l'ensemble des états finaux.

1.1. Automates finis (suite)

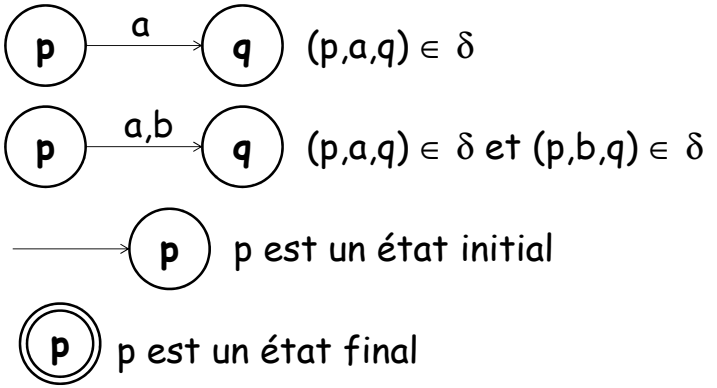
Définition formelle (suite)

δ : est une relation dans l'ensemble QxVxQ appelée aussi **relation de transition**.

(q_1,a,q_2) $\in \delta$ signifie que si l'automate se trouve dans l'état **q_1**, et le caractère à traiter est '**a**', alors l'automate peut passer à l'état **q_2**.

1.1. Automates finis (suite)

Représentation graphique



1.2. Chemin et trace

Un **chemin** de l'automate fini A=< Q, V, δ, I, F > de longueur **n** est une suite de transitions (p_i, a_{i+1}, p_{i+1}) avec $0 \geq i < n$

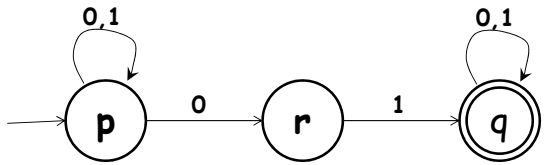
Ce chemin mène de l'état p_0 vers l'état p_n avec la **trace** $a_1 \ldots a_n$.

Remarque:

Le chemin de longueur 0 et de trace ε mène d'un état à lui-même.

1.2. Chemin et trace (suite)

Exp.



(p,1,p), (p,0,r), (r,1,q), (q,1,q) est un chemin de p à q avec la trace 1011

1.3. Mots et langages reconnus par un automate

Un mot x est reconnu par un automate s'il existe un chemin de trace x qui mène d'un état initial à un état final.

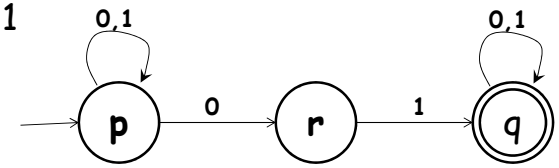
L(A) est le langage reconnu par l'automate A (tous les mots de L sont reconnus par A).

Remarque:

Deux automates sont **équivalents** s'ils reconnaissent le même langage.

1.3. Mots et langages reconnus par un automate (suite)

Exp. 1

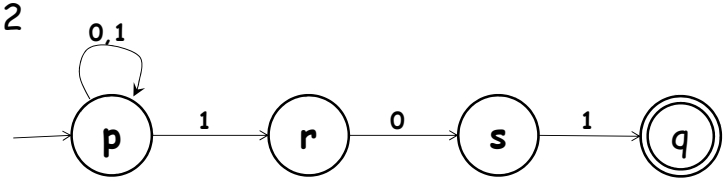


Le mot 01 est reconnu par cet automate.

Le langage reconnu est le langage contenant l'ensemble des mots sur {0,1} comportant le sous-mot 01.

1.3. Mots et langages reconnus par un automate (suite)

Exp. 2



Le langage reconnu cet automate est défini par l'expression régulière (0+1)*101, qui caractérise l'ensemble de mots sur l'alphabet {0,1} qui terminent par 101.

1.4. Automates finis déterministes

Un automate fini A = <Q, V, δ, I, F> est dit **déterministe** s'il n'a qu'un état initial, sans ε-transition, tel que pour tout état **p** ∈ Q et tout symbole **a** ∈ V, il y a **au plus un** état **q** tel que **(p,a,q)** ∈ δ

Comme l'ensemble **I** contient une seul état initial, il sera remplacé par cet état.

1.4. Automates finis déterministes (suite)

La relation δ peut être vue comme une fonction: $\delta : Q \times V \rightarrow Q$

$(p, a) \mapsto q = \delta(p, a)$

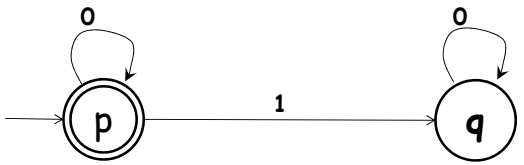
Une extension de cette fonction a été proposée:

$\delta^* : Q \times V^* \rightarrow Q$

$$\begin{cases} \delta^*(p, \varepsilon) = p \\ \delta^*(p, ax) = \delta^*(\delta(p, a), x) \quad \forall (p,a,x) \in Q \times V \times V^* \end{cases}$$

1.4. Automates finis déterministes (suite)

Exp.



1.4. Automates finis déterministes (suite)

On peut écrire aussi:

• Soit $A = \langle Q, V, \delta, i, F \rangle$ un automate déterministe. Le langage de cet automate est:

$\mathbf{L(A) = \{x \in V^* \mid \delta(i,x) \in F\}}$

• Pour tout état $\mathbf{p} \in Q$ et toutes chaînes $x,y \in V^*$, on a:

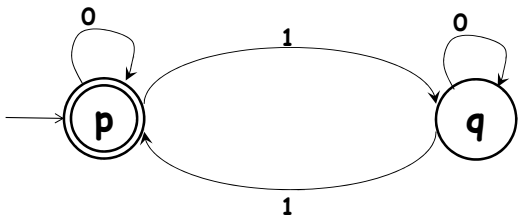
$\mathbf{\delta(p,xy) = \delta(\delta(p,x),y)}$

1.4. Automates finis déterministes (suite)

Un AEFD $A = \langle Q, V, \delta, i, F \rangle$ est dit **complet**, ou **complètement spécifié**, si :

Pour tout état $\mathbf{p} \in Q$ et tout symbole $\mathbf{a} \in V$, il y a **un et un seul** état **q** tel que $\mathbf{\delta(p,a) = q}$

Exp.



1.4. Automates finis déterministes (suite)

1.4.1. État accessible, automate connecté

Soit $A = \langle Q, V, \delta, i, F \rangle$ un AEFD.

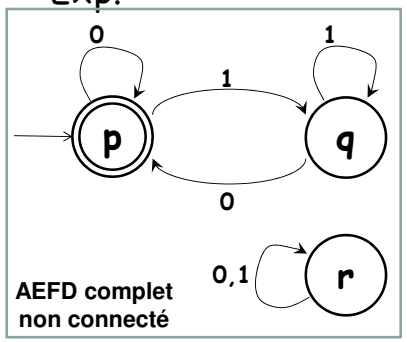
L'état $\mathbf{p} \in Q$ est dit **accessible** ssi il existe une chaîne $x \in V^*$, tel que: $\boldsymbol{\delta(i,x) = p}$

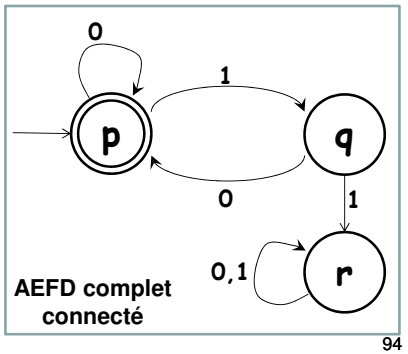
Un AEFD dont tous les états sont accessibles est dit **connecté**.

1.4. Automates finis déterministes (suite)

1.4.1. État accessible, automate connecté

Exp.





1.5. Minimisation des AEFD

Pour construire un AEFD équivalent à un autre mais avec un **nombre d'états réduit** on fait appel à un algorithme de minimisation appelé **Table-Filling algorithm**.

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling

Soit un AEFD $A = \langle Q, V, \delta, i, F \rangle$ et soit deux états $p,q \in Q$. C'est deux états sont dits **équivalents** ssi pour toute chaîne $x \in V^*$, $\boldsymbol{\delta(p,x) \in F \Leftrightarrow \delta(q,x) \in F}$

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

L'algorithme essai d'identifier les états non équivalents.

Les états qui restes sont **équivalents** et peuvent être **fusionnés**.

Pour cela, il faut suivre les deux règles suivantes:

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

- **Règle de base**: Soit p, q ∈ Q,

  Si (p ∈ F ∧ q ∉ F) ∨ (p ∉ F ∧ q ∈ F) alors p et q ne sont pas équivalents.

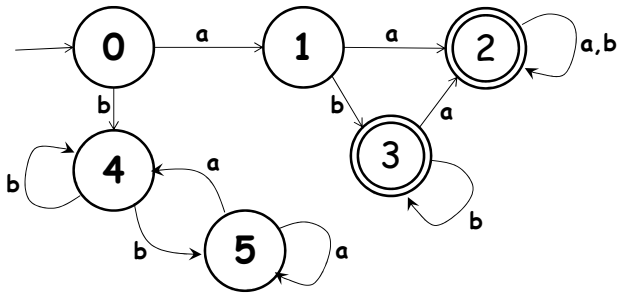
- **Règle par induction**: Soit p, q, r, s ∈ Q, a ∈ V,

  Si (r, s) = (δ(p,a), δ(q,a)) et r, s ne sont pas équivalents alors p, q ne sont pas équivalents.

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

Exécution de l'algorithme sur un exemple:



1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

	0	1	2	3	4	5
5						
4						
3						
2						
1						
0						

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

	0	1	2	3	4	5
5			X	X		
4			X	X		
3	X	X				
2	X	X				
1						
0						

Appliquons la règle de base.

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

	0	1	2	3	4	5
5			X	X		
4			X	X		
3	X	X				
2	X	X				
1						
0						

(0,1) (0,4) (0,5) (1,4) (1,5) (2,3) (4,5)

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

Appliquons la règle par induction:

(0,1): $(\delta(0,a), \delta(1,a)) = (1,2)$ Non équivalents $\Rightarrow$ (0,1) non équivalents

(0,4): $(\delta(0,a), \delta(4,a)) = (1,5)$ Inconnu $\Rightarrow$ notons (0,4) sous (1,5)

$(\delta(0,b), \delta(4,b)) = (4,4)$ Pas d'informations

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

	0	1	2	3	4	5
5			X	X		
4			X	X		
3	X	X				
2	X	X				
1	X					
0						

~~(0,1)~~ ~~(0,4)~~ (0,5) (1,4) <u>(1,5)</u> (2,3) (4,5)

(0,4)

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

(0,5): (δ(0,a), δ(5,a)) = (1,5) Inconnu $\Rightarrow$ notons (0,5) sous (1,5)

(δ(0,b), δ(5,b)) = (4,4) Pas d'informations

(1,4): (δ(1,a), δ(4,a)) = (2,5) Non équivalents $\Rightarrow$ (1,4) non équivalents

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

	0	1	2	3	4	5
5			X	X		
4		X	X	X		
3	X	X				
2	X	X				
1	X					
0						

~~(0,1)~~ ~~(0,4)~~ (0,5) (1,4) (1,5) (2,3) (4,5)

(0,4)

(0,5)

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

(1,5): (δ(1,a), δ(5,a)) = (2,5) Non équivalents $\Rightarrow$ (1,5) non équivalents $\Rightarrow$ (0,4) non équivalents $\Rightarrow$ (0,5) non équivalents

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

	0	1	2	3	4	5
5	X	X	X	X		
4	X	X	X	X		
3	X	X				
2	X	X				
1	X					
0						

~~(0,1)~~ ~~(0,4)~~ (0,5) (1,4) (1,5) (2,3) (4,5)

(0,4)

(0,5)

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

Reste à vérifier les paires (2,3) et (4,5)

(2,3): (δ(2,a), δ(3,a)) = (2,2) Pas d'informations

(δ(2,b), δ(3,b)) = (2,3) Inconnu et même paire.

(4,5): (δ(4,a), δ(5,a)) = (5,5) Pas d'informations

(δ(4,b), δ(5,b)) = (4,4) Pas d'informations

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

Deux cases dans la table restent **vides**, ce qui signifie que les deux paires correspondantes sont **équivalentes**:

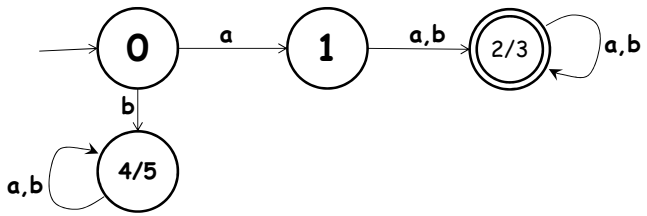
$$2 \equiv 3 \text{ et } 4 \equiv 5$$

Nous pouvons maintenant **fusionner** les états 2 et 3 ensemble et 4 et 5 ensemble.

1.5. Minimisation des AEFD (suite)

1.5.1. Algorithme Table-Filling (suite)

Après fusion:



2. Automates non déterministes et passage a un automate déterministe

2.1. Automates non déterministes

Un AEFND possède la même définition qu'un AEFD sauf que le retour de la fonction δ peut être un ensemble d'états.

Un automate fini A = <Q, V, δ, I, F> est dit **non déterministe** s'il n'a qu'un état initial, sans ε-transition, tel que pour tout état **p** $\in$ Q et tout symbole **a** $\in$ V, il y a **plusieurs** états $\mathbf{q_k}$ tel que $\mathbf{(p,a,q_k) \in \delta}$

2.1. Automates non déterministes (suite)

Définition formelle

Un AEFND est un quintuplet $A = \langle Q, V, \delta, i, F \rangle$ tel que:

δ: $Q \times V \rightarrow Q^*$

$(q, a) \mapsto \delta(q, a) = \{p_1, p_2, \ldots, p_n\}$

2.1. Automates non déterministes (suite)

Une extension de la fonction δ a été proposée:

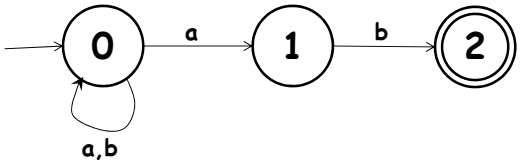
$\delta^* : Q \times V^* \rightarrow Q^*$

$$\begin{cases} \delta^*(p, \varepsilon) = p \\ \delta^*(p, ax) = \delta^*(\delta(p, a), x) \in Q^* \end{cases}$$

$\forall\ (p,a,x) \in Q \times V \times V^*$

2.1. Automates non déterministes (suite)

Exp:



$\delta^*(0, aabab) = \{0,2\}$

2.2. Langage d'un AEFND

Soit l'AEFND $A = \langle Q, V, \delta, i, F \rangle$.

La langage engendré par cet automate est :

$L(A) = \{w \mid \delta^*(i,w) \cap F \neq \emptyset\}$

Exp:

Le langage engendré par l'AEFND de l'exemple précédent est :

$L(A) = \{(a+b)^*ab \mid (a,b) \in V\}$

2.3. Passage d'un AEFN à un AEFD

Algorithme: Soit l'AEFND A = <Q, V, δ, i, F>.

1. Soit E_0 = {i}
2. Construire pour tout a∈ V:

   $E_1 = \delta(E_0,a) = \bigcup_{q\in E_0} \delta(q,a)$
3. Répéter l'étape 2 pour toute transition et pour chaque nouvel ensemble E_k
4. Tous les ensembles d'états E_k contenant au moins un état final deviennent des états finaux.

2.3. Passage d'un AEFN à un AEFD (suite)

Exp.

Soit l'AEFND suivant:

Etat	a	b
→ i	i, p2	p1
p1	p3	i, p2
* p2	p3, p4	p2
* p3	p2	p1
p4	∅	p3

2.3. Passage d'un AEFN à un AEFD (suite)

Exp. (suite)

Etat	a	b
i	i, p2	p1
i, p2	i, p2, p3, p4	p1, p2
p1	p3	i, p2
i, p2, p3, p4	i, p2, p3, p4	p1, p2, p3
p1, p2	p3, p4	i, p2
p3	p2	p1
p1, p2, p3	p2, p3, p4	i, p1, p2
p3, p4	p2	p1, p3
p2	p3, p4	p2

Etat	a	b
p2, p3, p4	p2, p3, p4	p1, p2, p3
i, p1, p2	i, p2, p3, p4	i, p1, p2
p1, p3	p2, p3	i, p1, p2
p2, p3	p2, p3, p4	p1, p2

2.3. Passage d'un AEFN à un AEFD (suite)

Exp. (suite) On renomme les ensembles d'états E_k

Etat	a	b
→ 0	1	2
*1	3	4
2	5	1
*3	3	6
*4	7	1
*5	8	2
*6	9	10
*7	8	11
*8	7	8

Etat	a	b
*9	9	6
*10	3	10
*11	12	10
*12	9	4

CHAPITRE IV
LES LANGAGES ALGÉBRIQUES

PLAN

1. Propriétés des langages algébriques
2. Les automates à pile

CHAPITRE IV
LES LANGAGES ALGÉBRIQUES

1. Propriétés des langages algébriques

Un langage algébrique ou hors-contexte est un langage engendré par une grammaire hors-contexte (type 2).

Rappel:

Une grammaire est dite **hors-contexte** si toutes ses règles sont du format suivant:

$u \rightarrow v$, tel que $u \in V_N$, $v \in V^*$

Le membre gauche de chaque règle est un seul symbole non terminal.

CHAPITRE IV
LES LANGAGES ALGÉBRIQUES

1. Propriétés des langages algébriques (suite)

Exp. Soit la grammaire $G = \langle V_T, V_N, S, R \rangle$

$R = \{S \rightarrow aSb, S \rightarrow ab\}$

G est une grammaire de type 2 (hors-contexte) qui engendre le langage algébrique: $\mathbf{a^n b^n}$

avec $n > 0$.

CHAPITRE IV
LES LANGAGES ALGÉBRIQUES

1. Propriétés des langages algébriques (suite)

1.1. Arbre de dérivation (syntaxique)

Dans une grammaire hors-contexte $G = \langle V_T, V_N, S, R \rangle$ on peut définir un arbre de dérivation pour une phrase du langage L(G), tel que:

- La racine est l'axiome;
- Chaque nœud interne représente un symbole non terminal $\in V_N$;
- Chaque feuille représente un symbole terminal $\in V_T$;

CHAPITRE IV
LES LANGAGES ALGÉBRIQUES

1. Propriétés des langages algébriques (suite)

1.1. Arbre de dérivation (syntaxique) (suite)

• Pour tout nœud interne représentant le non terminal **A** aillant **n** fils X_1, …, X_n, alors : ($A \rightarrow X_1X_2...X_n$) $\in$ R

Remarque:

La lecture de gauche à droite des feuilles de l'arbre reconstitue la phrase que l'arbre représente.

CHAPITRE IV
LES LANGAGES ALGÉBRIQUES

1. Propriétés des langages algébriques (suite)

1.1. Arbre de dérivation (syntaxique) (suite)

Exp: Soit $G = < V_T, V_N, S, R >$, tel que:

R = {S $\rightarrow$ ABA, A $\rightarrow$ CD, C $\rightarrow$ le|la, D $\rightarrow$ garçon|balle, B $\rightarrow$ frappe}

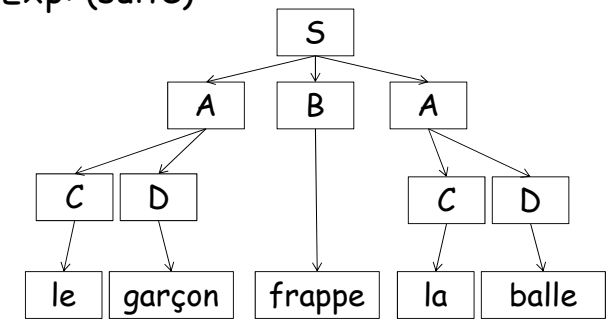
L'arbre de dérivation de la phrase: « le garçon frappe la balle » est comme suit:

CHAPITRE IV
LES LANGAGES ALGÉBRIQUES

1. Propriétés des langages algébriques (suite)

1.1. Arbre de dérivation (syntaxique) (suite)

Exp: (suite)



CHAPITRE IV
LES LANGAGES ALGÉBRIQUES

1. Propriétés des langages algébriques (suite)

1.2. L'ambigüité

Une **phrase** est dite **ambigüe** si elle accepte plus d'un arbre de dérivation.

Une **grammaire** est dite **ambigüe** si elle génère au-moins une phrase ambigüe.

1. Propriétés des langages algébriques (suite)

1.2. L'ambigüité (suite)

Exp. Soit $G = < V_T, V_N, S, R >$, tel que:

$V_T = \{x, +, -, *, /, (,)\}$

$V_N = \{S, A\}$

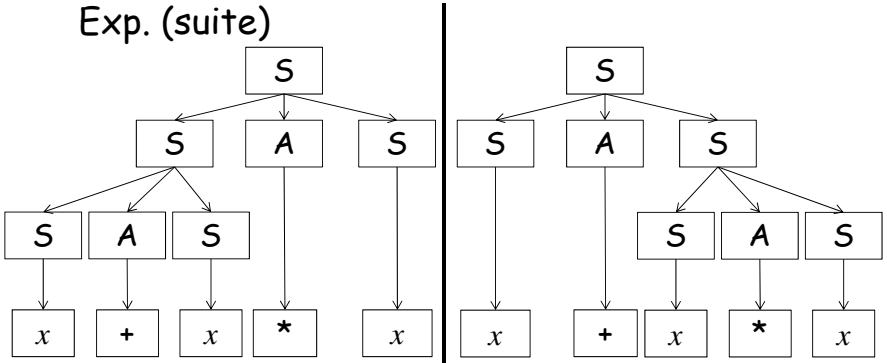
$R = \{S \rightarrow x|(S)|SAS,\ A \rightarrow +|-|*|/\}$

et soit les deux arbres suivants:

1. Propriétés des langages algébriques (suite)

1.2. L'ambigüité (suite)

Exp. (suite)



1. Propriétés des langages algébriques (suite)

1.2. L'ambigüité (suite)

Remarque:

On **ne peut pas** dire qu'un **langage est ambigu**, parce que souvent on peut transformer une grammaire ambigüe en une autre non ambigüe qui génère le même langage.

1. Propriétés des langages algébriques (suite)

1.3. La forme BNF d'une grammaire

L'écriture BNF (Backus-Naur Form) des règles de grammaire est définie depuis 1960 comme suit:

- Le symbole $\rightarrow$ est remplacé par ::=
- Les symboles terminaux sont écrits entre < et >
- On peut utiliser les parenthèses quand on a besoin. Exp: <A>::=(b|c)d ≡ <A>::=bd|cd

1. Propriétés des langages algébriques (suite)

1.3. La forme BNF d'une grammaire (suite)

• Un terminal contenant un seul caractère sera mit entre "a".

Exp. La forme BNF de la grammaire de l'exemple précédent:

<S> ::= "x"|"(" <S> ")"|<S> <A> <S>

<A> ::= "+"|"-"|"*"|"/"

1. Propriétés des langages algébriques (suite)

1.4. Propriétés de fermeture (ou clôture) des langages algébriques

D'une manière générale, un ensemble est **fermé** (ou **clôturé**) à une opération à **n** opérandes si tout résultat de cette opération appliquée à **n** éléments de l'ensemble **E** appartient aussi à l'ensemble **E**.

$$[a_1,\ldots, a_n] \in E^n : f(a_1, a_n) \in E$$

On dit que E est **fermé** à f.

1. Propriétés des langages algébriques (suite)

1.4. Propriétés de fermeture (ou clôture) des langages algébriques (suite)

Après la projection de cette notion sur l'espace de la théorie des langages, on remplace l'ensemble **E** par l'ensemble des langages algébriques et f par (union, produit, itération)

1.4. Propriétés de fermeture (ou clôture) des langages algébriques (suite)

Théorème:

L'**union** de deux langages algébriques est un langage algébrique, le **produit** (ou concaténation) de deux langages algébriques est un langage algébrique, et **l'itéré** d'un langage algébrique est un langage algébrique.

On peut dire que l'ensemble des langages algébriques est **fermé** aux opérations (union, produit, itération).

1.4. Propriétés de fermeture (ou clôture) des langages algébriques (suite)

Soit deux langages algébriques:

•L_1 *généré par la grammaire*

$$G_1 = \langle V_{T1}, V_{N1}, S_1, R_1 \rangle,$$

•L_2 *généré par la grammaire*

$$G_2 = \langle V_{T2}, V_{N2}, S_2, R_2 \rangle,$$

tel que: $V_{N1} \cap V_{N2} = \varnothing$

Sinon on renomme les non-terminaux en commun.

1.4. Propriétés de fermeture (ou clôture) des langages algébriques (suite)

•**L'union** $L_1 \cup L_2$ *est généré par la grammaire hors-contexte* $G = \langle V_T, V_N, S, R \rangle$, *tel que:*

$V_T = V_{T1} \cup V_{T2}$

$S \notin V_{N1} \cup V_{N2}$

$V_N = V_{N1} \cup V_{N2} \cup \{S\}$

$R = R_1 \cup R_2 \cup \{S \rightarrow S_1 | S_2\}$

1.4. Propriétés de fermeture (ou clôture) des langages algébriques (suite)

•Le **produit** ou la **concaténation** L_1L_2 *est généré par la grammaire hors-contexte* $G = \langle V_T, V_N, S, R \rangle$, *tel que:*

$V_T = V_{T1} \cup V_{T2}$

$S \notin V_{N1} \cup V_{N2}$

$V_N = V_{N1} \cup V_{N2} \cup \{S\}$

$R = R_1 \cup R_2 \cup \{S \rightarrow S_1S_2\}$

1.4. Propriétés de fermeture (ou clôture) des langages algébriques (suite)

•L'**itéré** L_1^* *est généré par la grammaire hors-contexte* $G = \langle V_T, V_N, S, R \rangle$, *tel que:*

$V_T = V_{T1}$

$S \notin V_{N1}$

$V_N = V_{N1} \cup \{S\}$

$R = R_1 \cup \{S \rightarrow S_1S | \varepsilon\}$

1.4. Propriétés de fermeture (ou clôture) des langages algébriques (suite)

Attention:

L'**intersection** de deux langages algébriques ne l'est pas nécessairement.

2. Automates à pile

Un automate à pile est un automate d'états fini avec une pile de capacité illimitée initialement vide.

Les langages algébriques sont reconnus par des automates à pile.

L'automate à pile consulte le sommet de sa pile et le remplace par une suite de symboles.

2. Automates à pile

Problématique:

Le langage $L = \{a^n b^n, n>0\}$ ne peut être reconnu par un automate d'états fini. Il faut utiliser un automate à pile.

2. Automates à pile (suite)

Définition formelle:

Un automate à pile est un 7-uplet: $AP = \langle Q, T, X, \delta, q_0, Z_0, F \rangle$

tel que:

Q: Ensemble fini d'états;

T: Vocabulaire terminal;

X: Vocabulaire de la pile (contient au-moins le symbole de fond de pile Z_0); on peut avoir $T \cap P = \varnothing$

2. Automates à pile (suite)

$q_0 \in Q$ est l'état initial;

F: Ensemble des états finaux $\subseteq Q$;

δ: Fonction de transitions définie comme suit:

$$\delta : Q \times (T \cup \{\varepsilon\})^* \times X^* \to \{Q \times X^*\}$$
$$(p, a, \alpha) \mapsto \{(q, \alpha')\}$$

tel que:

$\alpha \in X^*$; Contenu de la pile avant la transition.

$\alpha' \in X^*$; Contenu de la pile après la transition.

2. Automates à pile (suite)

Les actions possibles sur la pile sont:

- Dépiler le symbole au sommet de la pile;
- Empiler un symbole au sommer de la pile;

Important: Quand on écrit: **$(p,\beta) \in \delta(q,a,\alpha)$**

ça veut dire: Si on est à l'état 'q', et on rencontre le terminal 'a', et le sommet de la pile est α, alors on passe à l'état 'p', on dépile α de la pile et puis on empile β dans la pile.

2.1. Configuration de l'automate à pile

Une configuration est un triplet $(q, w, \alpha) \in Q \times (T \cup \{\varepsilon\})^* \times X^*$ qui représente l'état de l'automate à un instant donné.

q: est l'état actuel;

w: partie de la chaîne restante à lire;

α: Contenu de la pile.

2.1. Dérivation dans un automate à pile

Une dérivation dans l'automate à pile AP est un changement dans sa configuration. Elle est représentée par le symbole: $\vdash_{AP}$

On écrit:

$(q_1, aw, \alpha) \vdash_{AP} (q_2, w, \beta)$

Tel que:

$a \in T \cup \{\varepsilon\}$; $w \in T^*$; $q_1, q_2 \in Q$.

$\alpha, \beta \in X^*$; Contenus de la pile avant et après dérivation.

2.1. Dérivation dans un automate à pile (suite)

Exp. Soit la dérivation suivante:

$$(q_1, aw, \alpha\gamma) \vdash_{AP} (q_2, w, \beta\gamma)$$

Cette dérivation est faisable si on a:

$$(q_2, \beta) \in \delta(q_1, aw, \alpha)$$

2.1. Dérivation dans un automate à pile (suite)

Une succession de dérivations dans l'automate à pile AP est représentée par le symbole: $\vdash_{AP}^*$

On écrit:

$(q_1, w, \alpha) \vdash_{AP}^* (q_2, w', \beta)$

Tel que:

$a \in T \cup \{\varepsilon\}$; $w, w' \in T^*$; $q_1, q_2 \in Q$.

$\alpha, \beta \in X^*$; Contenus de la pile avant et après la succession des dérivations.

2.2. Configuration initiale d'un automate à pile

La configuration **initiale** de l'automate à pile AP = <Q, T, X, δ, q_0, Z_0, F> est:

$$\mathbf{(q_0, w, Z_0)}$$

avec $w \in T^*$

2.3. Configuration finale d'un automate à pile

Il existe 3 types de configurations **finales** de l'automate à pile AP = <Q, T, X, δ, q_0, Z_0, F> :

1. **Etat final**: **(q, ε, α)** avec $q \in F$ et $\alpha \in X^*$
2. **Pile vide**: **(q, ε, ε)** avec $q \in Q$
3. **Etat final et Pile vide: (q, ε, ε)** avec $q \in F$

Remarque:

Dans les 3 cas, le mots doit être lu entièrement.

2.4. Langage reconnu par un automate à pile

Le langage L(AP) $\subseteq T^*$ est reconnu par l'automate à pile AP par **état final** s'il est défini comme suit:

L(AP) = {w: w$\in T^*$ tel que:

$(q_0, w, Z_0) \vdash_{AP}^* (q, \varepsilon, \alpha)$, avec $q \in F$ et $\alpha \in X^*$ }

2.4. Langage reconnu par un automate à pile (suite)

Le langage $L_\varepsilon(AP) \subseteq T^*$ est reconnu par l'automate à pile AP par **pile vide** s'il est défini comme suit:

$L_\varepsilon(AP)$ = {w: w$\in T^*$ tel que:

$(q_0, w, Z_0) \vdash_{AP}^* (q, \varepsilon, \varepsilon)$, avec $q \in Q$}

2.4. Langage reconnu par un automate à pile (suite)

Théorème:

Un langage est algébrique (hors-contexte) si et seulement si il est accepté par un automate à pile.

Autrement dit:

Les langages reconnus par les automates à pile sont exactement les langages algébriques (hors-contexte).

Exemple:

Soit l'automate à pile AP = <Q, T, X, δ, 0, z, F> défini comme suit:

X = {y, z}, Q = {0, 1, 2}, T = {a, b}, F = {0}

δ est définie par le tableau:

(q,α)	$\delta(q,\varepsilon,\alpha)$	$\delta(q,a,\alpha)$	$\delta(q,b,\alpha)$
(0,z)	$\varnothing$	{(1,yz)}	$\varnothing$
(1,y)	$\varnothing$	{(1,yy)}	{(2,ε)}
(2,y)	$\varnothing$	$\varnothing$	{(2,ε)}
(2,z)	{(0,ε)}	$\varnothing$	$\varnothing$

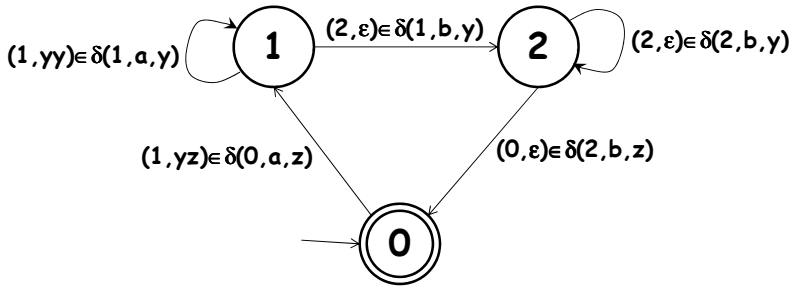
Exemple: (suite)

Rappel:

$(2, \beta) \in \delta(1, a, \alpha) \Leftrightarrow$ Si on est à l'état 1, et on rencontre le terminal 'a', et le sommet de la pile est α, alors on passe à l'état 2, on dépile α de la pile et puis on empile β dans la pile.

Exemple: (suite)

Le graphe de transition de AP est comme suit:



Exemple: (suite)

La configuration finale atteinte avec état final et Pile vide : **(0, ε, ε)**

Le langage reconnu par cet automate à pile est:

$L(AP) = \{(a^n b^n)^*, n \geq 0\}$