Couper le texte en mots (token) : Analyse lexical.

Expression à du sens ? Analyse Semantique

↳ Oubli declaration

↳ Taille de variable (int/long).

"#" ⇒ Préprocesseur

#include ... ⇒ remplace la ligne par le contenu du fichier.

Langage : Ensemble de mot (infini ou pas, pas necessairement d'order, peut être vide)

Mot : ~~Ensemble de~~ Suite finie de symbole, peut-être vide.

Suite ⇒ Ordre

Ensemble ⇏ Ordre

$\emptyset \neq \{\varepsilon\}$

Ensemble vide — Ensemble contenant le mot vide

Representation Ensemble ou Alphabet : $\Sigma_1 = \{a, b, c\}$

$\Sigma_2 = \{$☼, ⌂, ☁$\}$

Caracteristique Alphabet : nbre de symbole.

~~Langua~~

Alphabet ⊂ Language ⇒ Alphabet (non-vide, fini).

$\Sigma = \{a, b\}$

Σ^* : tous les mots que je peux former avec l'alphabet (infini)

$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \dots\}$.

$\forall u \in \Sigma,\ u \in \Sigma^*$ u est un mot

$\forall L \subset \Sigma^*$ L est un langage

Concatenation : associative, non-commutative,

$\left.\begin{array}{l} u = abc \\ v = bcd \end{array}\right\} u.v = abcbcd$.

Nombre de lettre

$|u| = 3$ $|u| + |v| = |u.v|$

Operation sur les langages : Union, Intersection, Complement

Concatenation : $\forall L_1 \subset \Sigma^*$; $\forall L_2 \subset \Sigma^*$

$L_1.L_2 = \{ab, \varepsilon, bc\} \cdot \{bc, \varepsilon, cd\} = \{abbc, abcd, bcbc, bccd,$ $bc, cd, \varepsilon\}$

Langage vide : Element absorbant

Langage $= \{\varepsilon\}$: Element neutre

Union :

$\forall n \in \mathbb{N},\ \forall u \in \Sigma^*\ \ \forall L \subset \Sigma^*$

$u^n = \begin{cases} u.u.u.u\dots \text{ (n fois)} \\ \varepsilon \text{ (n=0)} \end{cases}$

$L^n = \begin{cases} L.L.L\dots \text{ (n fois)} \\ \{\varepsilon\} \text{ (n=0)} \end{cases}$

Prefixe (mots) : $pref(abcaub) = \{\varepsilon, a, ab, abc, abca\}$

Prefixe langage : tous les prefixes de tous les mots du langage.

$L^* = \bigcup_{n=0}^{\infty} L^n = \underbrace{L^0}_{\{\varepsilon\}} \cup L \cup \underbrace{L^2}_{L.L} \cup L^3 \dots$

$L^+ = L.L^* = L^*.L = \bigcup_{n=1}^{\infty} L^n$

Base: $\emptyset, \{\varepsilon\}, \forall a \in \Sigma \{a\}, \cdot, \cup, *$ $\Rightarrow$ Langage Rationel $\Rightarrow$ $\emptyset \ . \ \varepsilon \ a \ | \ *$

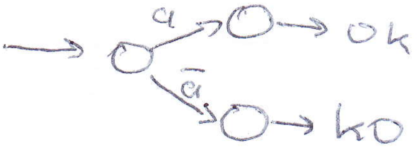
$\Sigma = \{-, +, 0, 1, 2, \dots, 9\}. \Rightarrow \{-\}.(\{0\} \cup \{1\} \cup \{2\} \dots \{9\})^+ \Leftrightarrow (+|-|\varepsilon)(0|1|2|3|\dots|9)^+$

Mot = Suite (finie / nulle) de symbole provenant d'un alphabet

$\emptyset \to \phi \to$ ○ ○→ False tout le tps

$\{\varepsilon\} \to \varepsilon \to$ ○ $\xrightarrow{\varepsilon}$ ○ True tout le tps

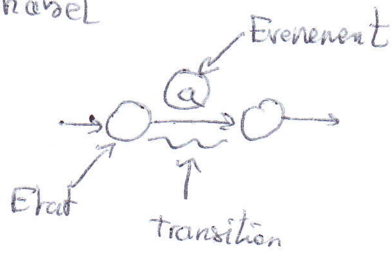
$\forall a \in \{a\} \to a \to$



$\cdot \to$

$\cup \to |$

$* \to *$



$(\Sigma, Q, I, F, \delta)$

Σ : alphabet de travail

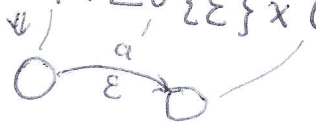
Q : Ensemble d'états

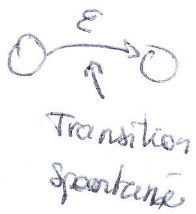
$I \subset Q$: Ensemble d'états initiaux

$F \subset Q$: Ensemble d'états finaux

δ : Ensemble de transition

$\hookrightarrow Q \times \Sigma \cup \{\varepsilon\} \times Q \Rightarrow \varepsilon$-NFA

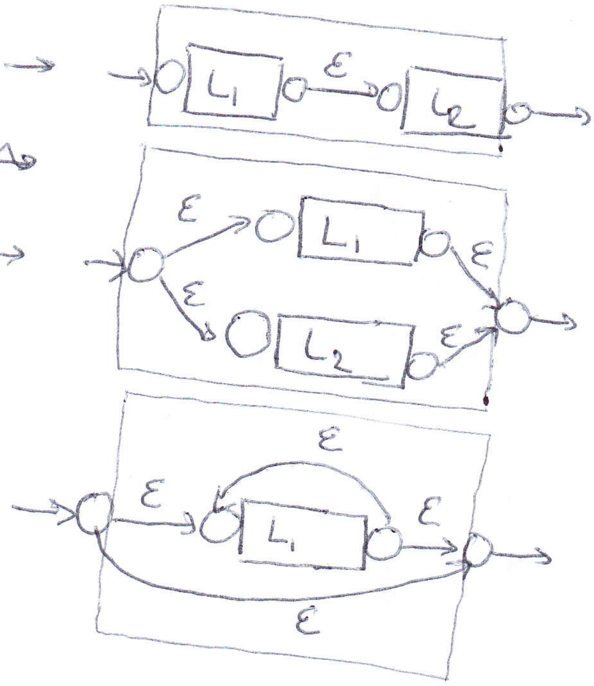




·

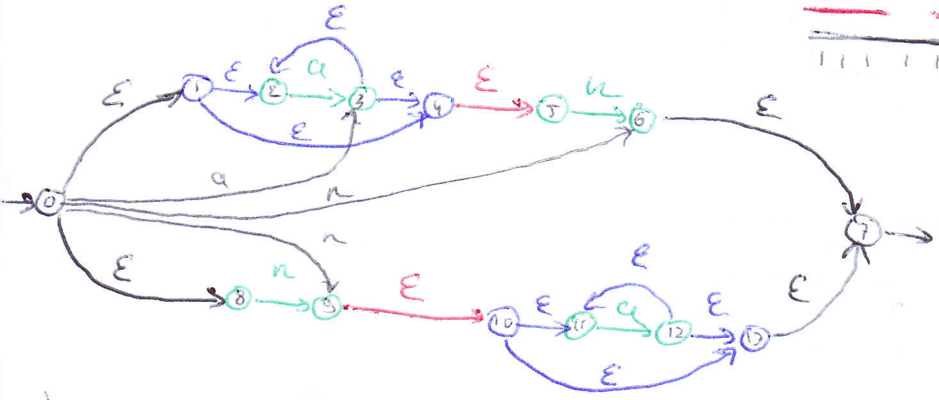
∪

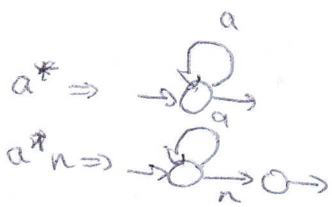
*



Construction de Thomson : + Automate à n états : $a^*n + na^*$

7 ⟹ 14 états.



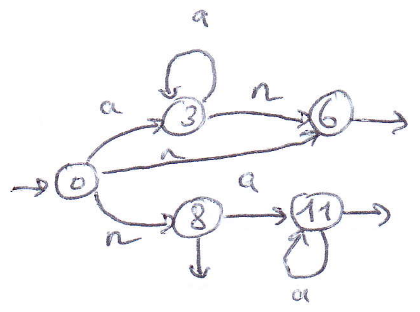


opti

ε f(i) 0 1 2 3 4 5 6 7 8 9 10

$\varepsilon - f(0) = \{0, 1, 7, 2, 4, 5\}$

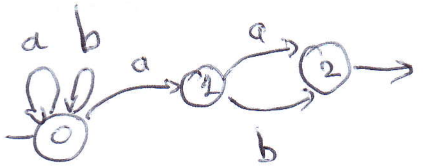
E(I)	1	2	3
0	0, 1, 7	0, 1, 2, 4, 7	0, 1, 2, 4, 5, 7
1	1, 2, 4	1, 2, 4, 5	1, 2, 4, 5
2	2	2	2
3	3, 2, 4	3, 2, 4, 5	3, 2, 4, 5
4	4, 5	4, 5	4, 5
5	5	5	5
6	6, 13	6, 13	6, 13
7	7	7	7
8	8, 9	8, 9, 10, 12	8, 9, 10, 12, 13
9	9, 10, 12	9, 10, 12, 13	9, 10, 12, 13
10	10	10	10
11	10, 11, 12	10, 11, 12, 13	10, 11, 12, 13
12	12, 13	12, 13	12, 13
13	13	13	13



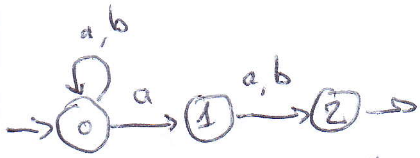
$(\Sigma, Q, I, F, \delta) = NFA$

$I \subseteq Q$, $F \subseteq Q'$, δ : $Q \times \Sigma \to Q$

Un etat utile : etat accessible ET co-accessible

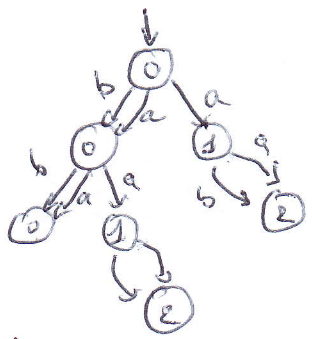


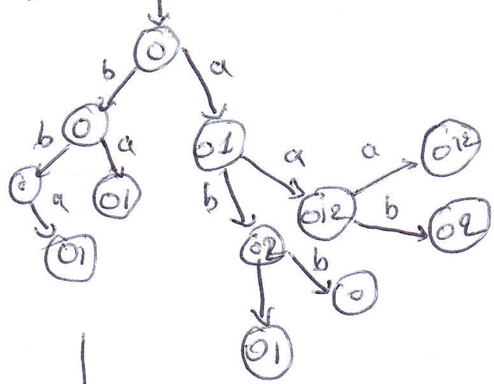
$(a+b)^* a (a+b)$



Complexité : exponentielle

Automate non-deterministe





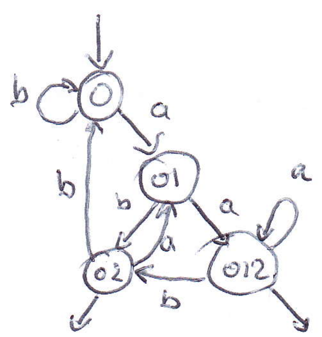
T	a	b
0	01	0
01	012	02
012	012	02
02	01	0

n etat (NFA)

⇓

2^n etat (DFA)

T	a	b
0	1	0
1	2	3
2	2	3
3	1	0



L. Rat → Exp Rat. → ε-NFA → NFA → DFA

Algo de recherche de Boyer-moore

Automate deterministe

Complexité : linéaire

$(\Sigma, Q, i, F, \delta)$, $Q \times \Sigma \to Q$

$i \in Q$, $F \subseteq Q$

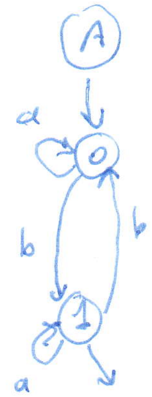
$\forall n \in \mathbb{N}$, $\Sigma = \{a, b\}$, $L\ a^n b^n$

C6 / 800
610 810

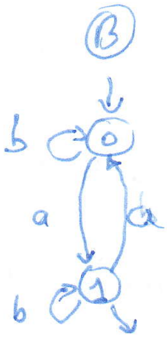
Pref(L₁) = ajouter des sorties sur les états coaccessible
préserve la rationalité

Suff(L₁) = ajouter des entrées sur les états accessible

Sous-mot(L₁) = Ajout d'entrées et sortie sur les états utiles.

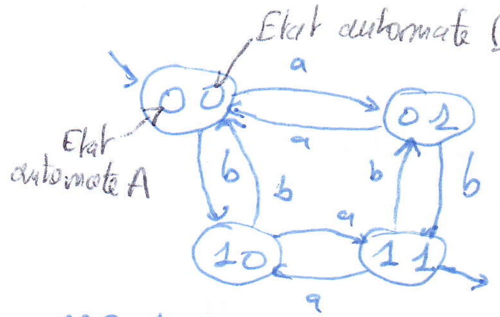


a impaire



b impaire

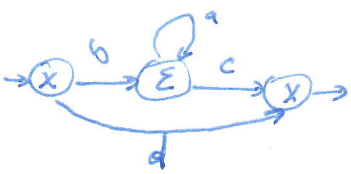
Intersection : Nbre de a et de b impaire.



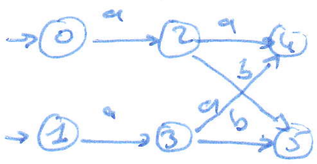
Produit synchronisé de 2 automates.

Si l'intersection est nulle, l'automate synchroniser n'aura pas de sortie.

BMC :



: $b(a^*)c + d$



	a	b
0	{1,3}	–
1	{2,3}	–

	a	b
4	–	–
5	–	.

	a	b
2		
3		

Grammaire : (N, Σ, P)

Alphabet Non-Terminaux
Alphabet Terminaux
Règle de Production

P → S V
S → 'il' | 'elle'
V → 'parle' | 'mange'

Element Non terminal

Contexte sensitive : $\underbrace{b}_{P}\ Q\ \underbrace{c}_{S} \rightarrow \underbrace{b}_{P}\ bc\ \underbrace{c}_{S}$ OK

$\underbrace{c}_{P}\ Q\ \underbrace{}_{} \rightarrow \underbrace{}_{\cancel{P}}\ Q\ \underbrace{c}_{}$ KO

KO : CQ → QC

ok : CQ → CX
CX → QX
QX → QC

Contexte Free : $\underset{\in N}{A} \to \underset{\in V^+}{\delta}$

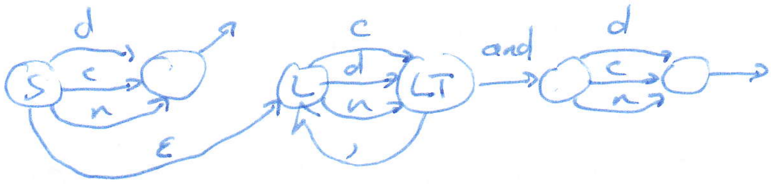
$\in N$ ← Non terminal ; $\in V^+$ ← (Non) terminal

Linéaire : $A \to v$
$A \to vB$
$B \to wC$ } Langage Rationel

$S \to c \mid d \mid n \mid L$

$L \to cLT \mid dLT \mid nLT$

$LT \to L \mid andc \mid anddl \mid andr$



(N, Σ, δ, S)
↓
$V^+ \times V^*$

Contrainte de Monotonie

$\to \gamma \to \alpha$

$|\gamma| \leq |\alpha|$

longueur.

Pour eviter que la regle ne consomme des caractères.

Contrainte sensitive.

$\underbrace{\gamma}_{V^*}\,A\,\underbrace{\alpha}_{V^*} \to \underbrace{\gamma}_{V^*}\,\underbrace{\delta}_{V^+}\,\underbrace{\alpha}_{V^*}$

Reecrire que la partie non terminal.

⇒ Un symbol terminal reste jusqu'à la fin (cas d'arrêt)

Linéaire

$A \to v\vec{B}$

$B \to v\vec{C}$

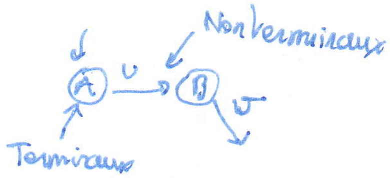
$C \to w\vec{D}$

} → L. Rationel.

Non linéaire

$A \to v\vec{B}$

$B \to \overleftarrow{C}v$



Terminaux = état

Non-terminaux = transition

$A \to \underbrace{\gamma}_{\in \Sigma^+}$ } Langage fini

Grammaire

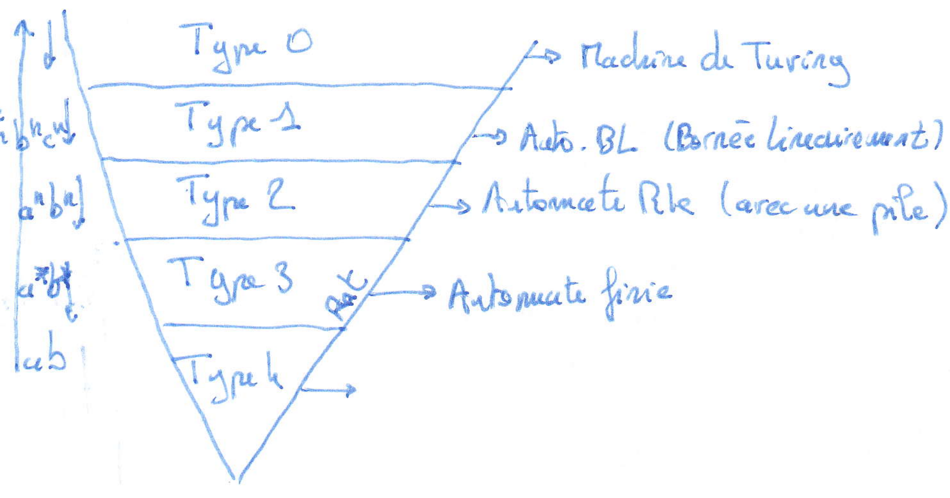
Type 0 : aucune contrainte.

Type 1 : Monotonie, Contexte-Sensitive

Type 2 : Contexte Free

Type 3 : Linéaire droite et gauche
↳ Rationnel.

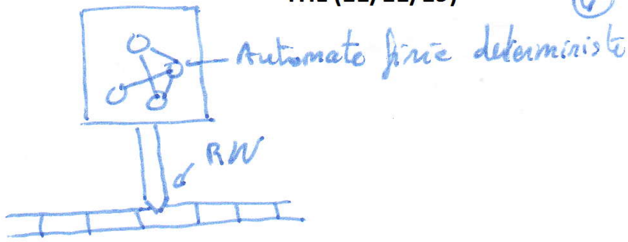
Type 4 : Choix fini.



Le plus gd lang : Σ^* (type 3)

Plus on monte plus on gagne en expressivité.

Machine de Turing :



Memoire infinie :

Algorithme raisonnable : complexité polynomiale.

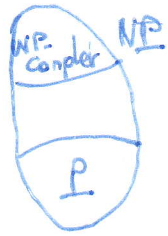
NP $\stackrel{?}{=}$ P

NP : Non deterministe.

P : Ensemble des algos bornée polynomiale avec un automate deterministe à état finie.

P ⊂ NP [OK]

⊃ ?



On peut passer d'un problème NP-Complet à un autre via un algo polynomiale.

$\mathcal{A}^1 \propto \mathcal{A}^2 \propto \mathcal{A}^3$. Si on en resoud un, on les resoud tous (en temps raisonnable)

<A> ::= <B> c

---

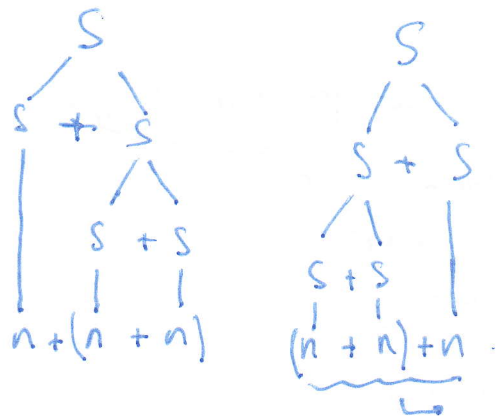
1. $S \rightarrow S + S$
2. $S \rightarrow n$

Non linéaire Context Free. ▷ Type 2.

↳ 1 seul non terminal.

$S \xrightarrow{1} S + S \xrightarrow{2} 'n' + S \xrightarrow{1} n + S + S \xrightarrow{2} n + n + S \xrightarrow{2} n + n + n$

$S \xrightarrow{1} S + S \xrightarrow{2} S + n \xrightarrow{1} S + S + n \xrightarrow{2} n + S + n \xrightarrow{2} n + n + n$



Meme operateur : Associativité à gauche.

L'ordre est important en semantique. Autrement deux arbres possibilité signifie qu'il y a ambiguité.

$E \to E_+ \mid E_n$
$E_+ \to E + E$
$E_n \to n$

$E_+ \to E + E$

$E_+ + E_n$ ok, priorité à gauche
$E_n + E_+$ ko, parenthesage droit
$E_n + E_n$ ko
$E_+ + E_+$ ok

$E \to E_+ \mid E_n$
$E_+ \to E + E_n \mid E_n + E_n$
$E_n \to n$

$E \to E_+ \mid E_n$
$E_+ \to E + E_n$
$E_n \to n$

$E \to E + E_n \mid E_n$
$E_n \to n$

$E \to E_+ \mid E_* \mid E_n$
$E_+ \to E + E$
$E_* \to E * E$
$E_n \to n$

E_+

$E_+ + E_+$ ko
$E_+ + E_*$ ok
$\boxed{E_+ + E_n}$ ok
$E_* + E_+$ ko
$E_* + E_*$ ok
$\boxed{E_* + E_n}$ ok
$E_n + E_+$ ko
$E_n + E_*$ ok
$\boxed{E_n + E_n}$ ok

On garde tout ceux qui finissent par E_n car le parenthesage gauche est respecté.

$E_+ \to \underbrace{E_+ + E_*}_{F} \mid E_+ + E_n \mid E_* + E_* \mid \underbrace{E_* + E_n}_{F} \mid E_n + E_* \mid E_n + E_n$.

$\bigcirc = E$

E_*

$E_+ * E_+$ ko
$E_+ * E_*$ ko
$E_+ * E_n$ ko
$E_* * E_+$ ko
$E_* * E_*$ ko
$E_* * E_n$ ok
$E_n * E_+$ ko
$E_n * E_*$ ko
$E_n * E_n$ ok

$E_* \to E_* * E_n \mid E_n * E_n$

$F \to E_* \mid E_n$
$E \to E_+ \mid F$
$F \to F * E_n$
$E_+ \to E + F$

ccl

$E \to E + F \mid F$
$F \to F * E_n \mid E_n$
$E_n \to n \mid (E) \mid sqrt(E) \mid -E$

Raccourci : prio.

$E \to E + F$
$E \to F$
$F \to F * E_n$
$F \to E_n$
$E_n \to n$

THL 5

LL(k)

↑ ← Non-terminal le + à gauche en premier.

sens de lecture

inst → 'if' exp 'then' inst 'fi'

inst → 'print' 'foo'

exp → 'true' | 'false'

LL(k) LL(1)	if	true	false	then	fi	print
inst	1	/	/	/	/	2
exp	/	3	4	/	/	/