



www.dirasats.com



هذا الغلاف لا يعبر عن حقوق الملكية او فحوى الكتاب, فهو مجرد واجهة للموقع المحمل منه

شكرا لك على ثقتك بنا وعلى اختيار موقعنا



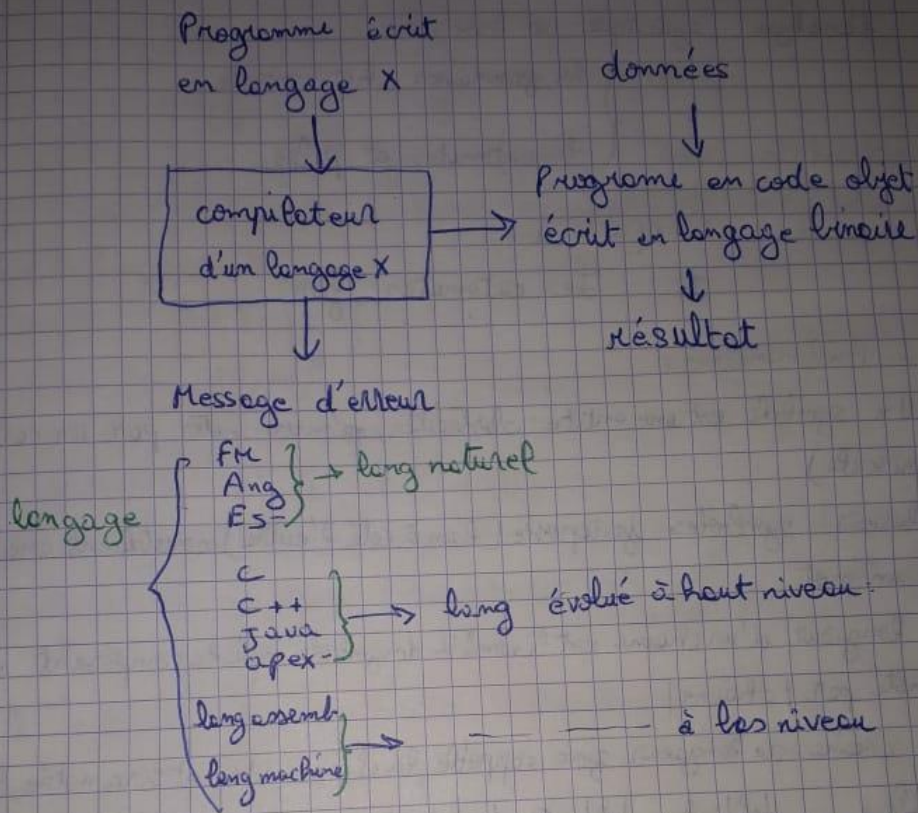
www.dirasats.com



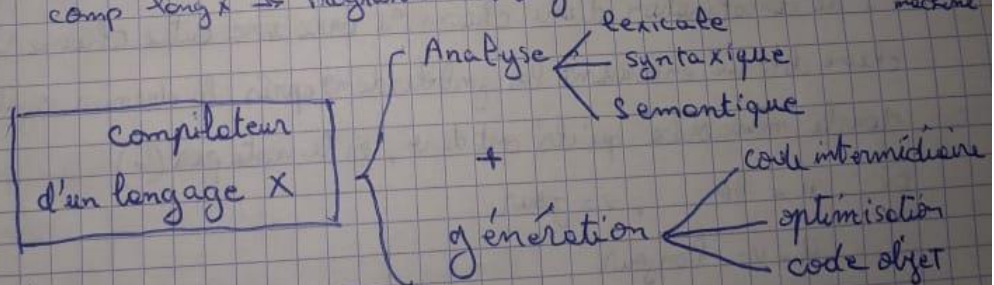
من اجل تواصل معنا المرجو زيارة الموقع ستجد جميع المعلومات

www.dirasats.com

Cours de Compilation



comp lang X → Programme écrit lang orienté assem → Prog en langage machine



flex : outils pour analyse lexicale

yacc : " " " " syntaxique

l'analyse lexicale est basé sur
{ les automates finis \Rightarrow ^{verif valide} ^{verif code} ^{verif code} ^{verif code} ^{verif code}
et
les expressions régulières

l'analyse syntaxique se base sur :
{ les grammaires, hors contexte
les automates à piles

Chapitre 1 :

Les automates finis

1. Préliminaires :

- Un symbole est une entité abstraite, qu'on va noter par un caractère (minuscule)
- Plusieurs symboles juxtaposés (l'un à côté l'autre) constituent une chaîne ou un mot
- la longueur d'une chaîne est le nombre de symboles qui la composent elle sera noté par $|chaîne|$
- La chaîne de longueur zéro s'appelle la chaîne vide et sera notée ϵ

Exemple : $|i| = 1$, $|\epsilon| = 0$, $|ababab| = 5$, $|au\text{ revoir}| = 9$

- La concaténation d'une chaîne u avec une autre chaîne v est la chaîne obtenue en écrivant les symboles de v après la dernier symbole de u dans le même ordre qu'ils ont de v , on le note par (\cdot)

Ex : $u = \text{bon}$, $v = \text{jour}$

$u \cdot v = \text{bonjour}$

$v \cdot u = \text{jourbon}$

Rq : la concaténation est une loi non commutative, elle admet ϵ comme élément neutre

$$u \cdot \epsilon = \epsilon \cdot u = u, \forall u$$

$$|u \cdot v| = |u| + |v|$$

chaîne	Prefixe	Suffixe
bonjour	ϵ b bo bon bons bonjo bonjou bonjour	bonjour onjour njour jour our ur r ϵ

Un alphabet est un ensemble fini de symboles, si Σ est un alphabet on peut définir Σ^i (où $i \in \mathbb{N}$) l'ensemble des chaînes composées à partir des symboles de Σ et dont la longueur est i

Par exemple Pour $\Sigma = \{0, 1\}$

$$\Sigma^2 = \{00, 01, 10, 11\}$$

$$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 111, 110\}$$

par convention $\Sigma^1 = \Sigma$ et $\Sigma^0 = \{\epsilon\}$

l'ensemble de toutes les chaînes possibles composées de symboles de Σ sera noté Σ^*

et s'appelle l'étoile de Kleene autrement écrit $\Sigma^* = \bigcup_{i \in \mathbb{N}} \Sigma^i$ on note $\Sigma^+ = \bigcup_{i \geq 1} \Sigma^i$

$$Rq: \Sigma^+ = \Sigma^* \cup \{\epsilon\}$$

$$\Sigma^{i+1} = \Sigma^i \cdot \Sigma \quad \text{les concaténation entre les ensembles}$$

Un langage (formel) sur un alphabet est un ensemble fini ou infini de chaînes de Σ^*

II. Automates finis Déterministes "deterministic finite Automata"

1) Définition:

Une automate fini déterministe (DFA) est la donnée de 5 uplet $(Q, \Sigma, \delta, q_0, F)$

Q : ensemble fini des états

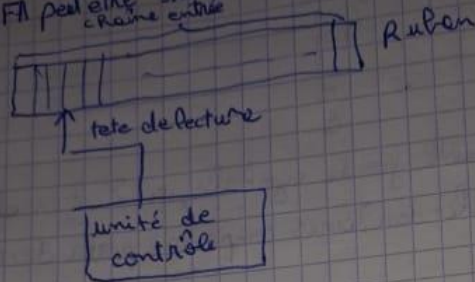
Σ : un alphabet (ensemble fini de symboles)

δ : fonction de transition

$$Q \times \Sigma \longrightarrow Q$$

$$(q, \Delta) \longmapsto \delta(q, \Delta)$$

q_0 état initial $\in Q$
 F ensemble des états finaux $\subset Q$
 Un DFA peut être schématisé par :



Exemple :

- L'ascenseur
- Le guichet longueur automatique
- Le distributeur des boissons automatique

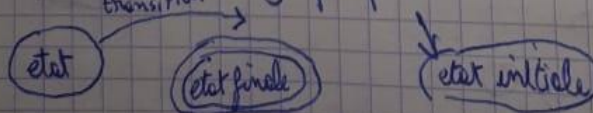
$$A_n = (Q, \Sigma, \delta, q_0, F)$$

$$\begin{cases} Q = \{q_0, q_1, q_2\} \\ \Sigma = \{0, 1\} \\ F = \{q_2\} \end{cases}$$

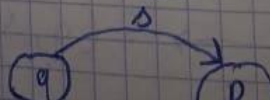
et δ donnée par la table

δ	0	1
q_0	q_1	q_2
q_1	q_0	q_2
q_2	q_1	q_2

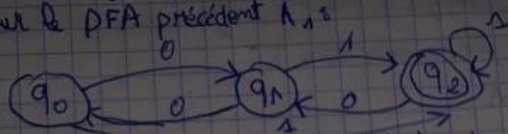
on peut présenter graphiquement un DFA



$$\text{si } \delta(q, a) = p$$

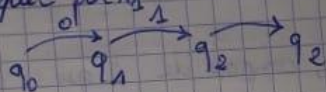


si $\delta(q, \epsilon) = q$
pour le DFA précédent A_1

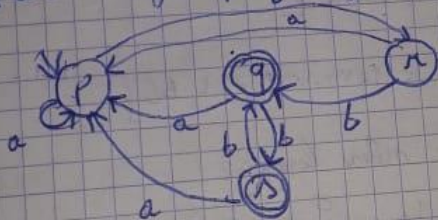


si la chaîne est de longueur 0 l'automate A_1 passe séquentiellement par les états $q_0, q_2, q_1, q_2, q_1, q_0$

Après l'examen de ~~chaîne~~ ^{dernier} symbole A_1 se trouve dans l'état q_0 qui n'est pas final. On dit alors que la chaîne 10100 n'est pas acceptée par A_1 .
Par contre l'examen de la chaîne 011 amène à l'état final q_2 on dit que cette chaîne est acceptée par A_1 .



A_2 le DFA définie par son graphe



La chaîne $bb, babb, aababbb$ sont acceptées par A_2
Par contre a, ab, baa, bab ne sont pas acceptés

$bb : p \rightarrow q$

$babb : p \rightarrow p \rightarrow q$

$aababbb : p \rightarrow p \rightarrow p \rightarrow q \rightarrow s$

Pour exécuter un DFA $(Q, \Sigma, \delta, q_0, F)$ sur toutes les chaînes de Σ^* ,
on propose l'extension suivante

$$\hat{\delta} : Q \times \Sigma^* \rightarrow Q$$

$$(q, w) \mapsto \hat{\delta}(q, w)$$

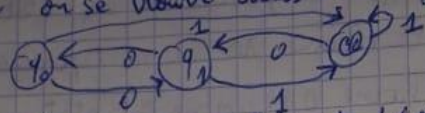
avec

$$\hat{\delta}(q, w) = \begin{cases} q & \text{si } w = \epsilon \\ \delta(\hat{\delta}(q, u), s) & \text{si } w = us \\ & u \in \Sigma^* \text{ et } s \in \Sigma \end{cases}$$

M.B. : $\hat{\delta}^n = \delta$ sur $Q \times \Sigma$
 car $\hat{\delta}(q, s) = \delta(\hat{\delta}(q, s), \epsilon) = \delta(q, s)$
 Dans la suite on note $\hat{\delta}$ par δ sur $Q \times \Sigma^*$

2) langage accepté par un DFA

On appelle langage accepté par un DFA $(Q, \Sigma, \delta, q_0, F)$ l'ensemble des chaînes de Σ^* acceptées par cet automate (après l'examen du dernier symbole on se trouve dans un état final)



Il sera noté $L(M)$ formellement $L(M) = \{w \in \Sigma^* \text{ telle que } \delta(q_0, w) \in F\}$

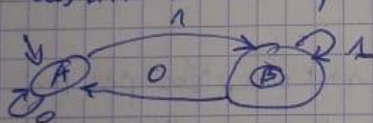
Exemple :

$L(A_1) = \{ \text{chaînes, composées de 1 qui se terminent par 1} \}$

$= \{ \text{entiers impaire codée en binaire} \}$

$L(A_2) = \{ \text{chaînes de } \{a, b\} \text{ qui se terminent par } bb \}$

Rq : des DFA suivent acceptent le même langage



- On dit que ces DFA sont équivalents

- tout langage accepté (ou reconnu) par un DFA s'appelle un langage régulier

DFA acceptent les mots réservés de C (program, if, while, do, else, file, for, switch)

Remarque : Tous les sous-ensembles finis de Σ^* sont des langages réguliers

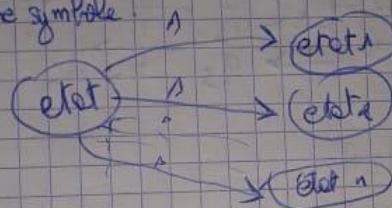
Ex : $\Sigma = \{a, b, c\}$

Normalement, de chaque état le nombre des transitions sortantes est égal au cardinal de l'alphabet. Dans un tel cas on dit que le DFA est complet.

On peut rencontrer des DFA non complet c-à-d $\delta(q, a)$ n'est pas toujours défini. Si au cours de l'examen d'une chaîne on a besoin d'une telle transition alors cette chaîne ne sera pas acceptée.

Théorème : Pour chaque langage régulier il existe un nombre minimal des états unique que le DFA qui l'accepte.

III - Automates finis non déterministe : "Non-deterministe finite Automate" NFA c'est un automate fini qui autorise plusieurs transitions à partir d'un état avec le même symbole.



$$\delta(\text{état}, a) = \{\text{état1}, \dots, \text{étatn}\}$$

1) Définition Formelle :

un NFA est la donnée d'un quintuplet $(Q, \Sigma, \delta, q, F)$ où Q, Σ, q et F ont les mêmes significations dans la def d'un DFA et δ la fonction de transition définie

$$\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$$

$$(q, a) \rightarrow \delta(q, a)$$

Exemple: $M_0(\{q_1, q_2, q_3\}, \{a, b\}, \delta, \{q_1\}, \{q_3\})$

avec :

$$\delta(q_1, a) = \{q_1, q_2, q_3\}$$

$$\delta(q_1, b) = \{q_2, q_3\}$$

$$\delta(q_2, a) = \emptyset$$

$$\delta(q_2, b) = \{q_2, q_3\}$$

$$\delta(q_3, a) = \emptyset$$

$$\delta(q_3, b) = \{q_3\}$$

$$\begin{aligned} \delta: Q \times \Sigma &\rightarrow P(Q) \\ (q, a) &\rightarrow \hat{\delta}(q, a) \end{aligned}$$

$$\hat{\delta}(q, \epsilon) = q$$

$$\hat{\delta}(q, yx) = \bigcup_{p \in \hat{\delta}(q, y)} \delta(p, x)$$

$$x = y\Delta \quad \text{avec } \Delta \in \Sigma \text{ et } y \in \Sigma^*$$

$$\text{ex: } \hat{\delta}(q_1, aab) = \bigcup_{p \in \hat{\delta}(q_1, aa)} \delta(p, b)$$

$$\hat{\delta}(q_1, aa) = \bigcup_{p \in \hat{\delta}(q_1, a)} \delta(p, a)$$

$$\hat{\delta}(q_1, a) = \bigcup_{p \in \hat{\delta}(q_1, \epsilon)} \delta(p, a) = \delta(q_1, a) = \{q_1, q_2, q_3\}$$

$$\begin{aligned} \Rightarrow \hat{\delta}(q_1, a a) &= \bigcup_{p \in \hat{\delta}(q_1, a)} \delta(p, a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \\ &= \{q_1, q_2, q_3\} \cup \emptyset \cup \emptyset \end{aligned}$$

$$\begin{aligned} \Rightarrow \hat{\delta}(q_1, a a b) &= \bigcup_{p \in \{q_1, q_2, q_3\}} \delta(p, b) = \delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b) \\ &= \{q_2, q_3\} \cup \{q_2, q_3\} \cup \{q_3\} \end{aligned}$$

Rq: $\hat{\delta}$ coïncide avec δ avec δ sur $Q \times \Sigma$ en même temps alors $\hat{\delta}$ est écrit pour le prolongement

On peut définir le langage accepté par un NFA M par $L(M) = \{x \in \Sigma^* / \delta(q_0, x) \in F\}$

$$L(M) = \{x \in \Sigma^* / \delta(q_0, x) \cap F \neq \emptyset\}$$

Théorème 2: Pour tout langage accepté par un NFA, il existe un DFA l'accepte. Autrement dit: Tout NFA est équivalent à un DFA

Démo: Comment trouver un DFA à partir d'un NFA $M = (Q, \Sigma, \delta, q_0, F)$

Pour cela, considérons le prolongement de δ défini par $\hat{\delta}: \Sigma^* \times P(Q) \rightarrow P(Q)$

$$\hat{\delta}(P, a) = \bigcup_{q \in P} \delta(q, a) \quad \text{pour tout } P \subset Q$$

$$\hat{\delta}(\{q_0\}, 11) = \varepsilon\text{-fermeture}(\delta(\{q_1, q_2\}, 1)) \\ = \varepsilon\text{-fermeture}(\{q_1\})$$

$$\hat{\delta}(\{q_0\}, 11) = \{q_1, q_2\}$$

$$\hat{\delta}(\{q_0\}, 012) = \varepsilon\text{-fermeture}(\delta(\hat{\delta}(q_0, 01), 2)) = ?$$

$$\hat{\delta}(q_0, 01) = \varepsilon\text{-fermeture}(\delta(\hat{\delta}(q_0, 0), 1)) = ?$$

$$\hat{\delta}(q_0, 0) = \varepsilon\text{-fermeture}(\delta(\hat{\delta}(q, \varepsilon), 0)) = \varepsilon\text{-fermeture}(\delta(\varepsilon\text{-fermeture}(q_0), 0)) \\ = \varepsilon\text{-fermeture}(\delta(\{q_0, q_1, q_2\}, 0)) = \varepsilon\text{-fermeture}(\{q_0\}) = \{q_0, q_1, q_2\}$$

$$\Rightarrow \hat{\delta}(q_0, 01) = \varepsilon\text{-fermeture}(\delta(\{q_0, q_1, q_2\}, 1)) \\ = \varepsilon\text{-fermeture}(\{q_1\}) = \{q_1, q_2\}$$

$$\Rightarrow \hat{\delta}(q_0, 012) = \varepsilon\text{-fermeture}(\delta(\{q_1, q_2\}, 2)) = \varepsilon\text{-fermeture}(\{q_2\}) = \{q_2\}$$

les langages acceptés par le NFA- ε $M = (Q, \Sigma, \delta, q_0, F)$ est l'ensemble :

$$L(M) = \{w \in \Sigma^* / \hat{\delta}(q_0, w) \cap F \neq \emptyset\}$$

$$\text{N.B. } \hat{\delta}(q, \varepsilon) \neq \delta(q, \varepsilon)$$

$$\text{On a obtenu que } \hat{\delta}(q_0, 0) = \{q_0, q_1, q_2\}$$

$$\text{et à partir du NFA-}\varepsilon \quad \delta(q_0, 0) = \{q_0\}$$

Théorème 1.0. (l'élimination des ε -transitions)

Pour tout langage accepté par un NFA- ε il existe un NFA qui l'accepte

En effet : soit $M = (Q, \Sigma, \delta, q_0, F)$ en NFA- ε qui accepte le langage

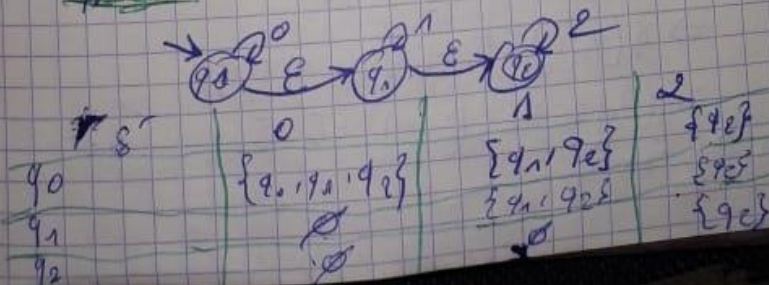
Considérons le NFA $M' = (Q', \Sigma, \delta', q'_0, F')$

$$\text{où } Q' = Q, \quad q'_0 = q_0$$

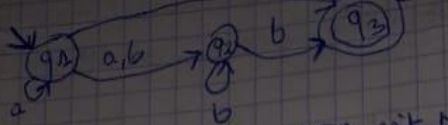
$$F' = \begin{cases} F \cup \{q_0\} & \text{si } F \cap \varepsilon\text{-fermeture}(q_0) \neq \emptyset \\ F & \text{sinon} \end{cases}$$

$$\delta'(q, a) = \delta(\{q\}, a) \quad \forall q \in Q \text{ et } \forall a \in \Sigma$$

Application : élimination des ε -transitions de



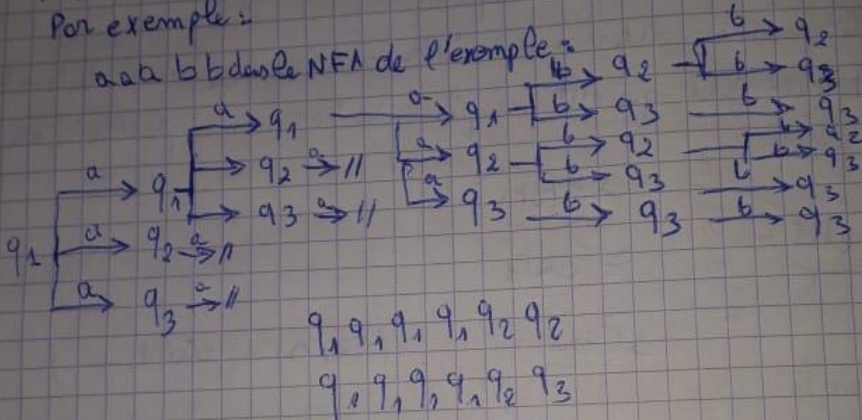
Le graphe de M sera a



2) Langage acceptés par NFA soit $M = (Q, \Sigma, S, q, F)$ un NFA; et soit $w = a_1 a_2 \dots a_n \in \Sigma^*$, On dit que w est acceptée par M s'il existe une séquence de transitions étiquetées respectivement par les symboles a_1, a_2, \dots, a_n qui commence de q et se terminent dans l'état de F

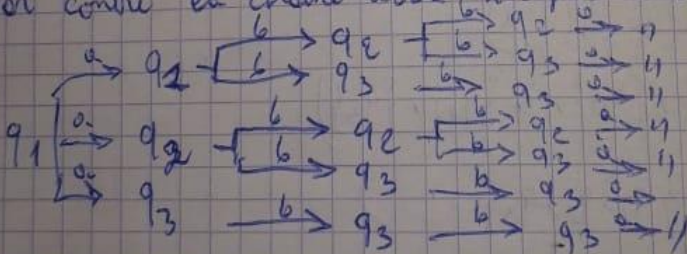
Par exemple :

aaa b dans le NFA de l'exemple :



La séquence $q_1 q_1 q_1 q_2 q_2 q_3$ se termine par $q_3 \in F$ donc aaa bb est accepté

par contre la chaîne abba n'est pas acceptée



L'ensemble de toutes les chaînes acceptées par un NFA s'appelle le langage accepté par cet automate

Pour formaliser cette application, considérons le prolongement de S suivant :



donc 001 est accepté par ce NFA- ϵ
 Par contre la chaîne 100 n'est pas acceptée (Avez-vous)
 L'ensemble de toutes les chaînes acceptées par un NFA- ϵ est un langage
 accepté par ce NFA- ϵ

Formalisation Définition des ϵ -fermetures

Soit $M = (Q, \Sigma, \delta, q, F)$ un NFA- ϵ et $q \in Q$
 On appelle ϵ -fermetures (q) l'ensemble des états accessibles
 par les ϵ -transitions à partir de q

Rq: ϵ -fermeture (q) n'est jamais vide car il contient au moins l'état q
 Par exemple dans le NFA- ϵ déjà donnée ϵ -fermeture = $\{q_0, q_1, q_2\}$

ϵ -fermeture (q_1) = $\{q_1, q_2\}$ ϵ -fermeture (q_2) = $\{q_2\}$

$X \subset Q$ on définit ϵ -fermeture(X) = $\bigcup_{p \in X} \epsilon$ -fermeture(p)

Prolongement de la fonction de transition δ

$$\hat{\delta} : P(Q) \times \Sigma^* \rightarrow P(Q)$$

$$\hat{\delta}(\{p\}, \epsilon) = \epsilon\text{-fermeture}(\{p\})$$

$$\hat{\delta}(\{p\}, x) = \epsilon\text{-fermeture}(p)$$

$$x \in \Sigma^* \text{ et } \Delta \in \Sigma$$

$$\text{où } P = \{q \in Q / \exists x \in \Sigma^* \text{ et } q \in \delta(\{p\}, x) \text{ et } q \in \delta(x, \Delta)\}$$

Exemple : Dans le NFA- ϵ précédent:

$$\hat{\delta}(\{q_0\}, \Delta) = \epsilon\text{-fermeture}(\{q_0\}) = \{q_0, q_1, q_2\}$$

calculons d'abord $\hat{\delta}(\{q_0\}, \Delta)$

$$\hat{\delta}(\{q_0\}, \Delta) = \epsilon\text{-fermeture}(\{q_0\}) = \{q_0, q_1, q_2\}$$

$$= \epsilon\text{-fermeture}(\{q_0\}) = \{q_0, q_1, q_2\}$$

$$= \epsilon\text{-fermeture}(\{q_0\}) = \{q_0, q_1, q_2\}$$

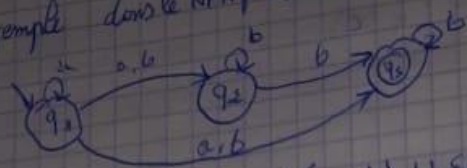
$$= \epsilon\text{-fermeture}(\{q_0\}) = \{q_0, q_1, q_2\}$$

$$= \epsilon\text{-fermeture}(\{q_0\}) = \{q_0, q_1, q_2\}$$

$$= \epsilon\text{-fermeture}(\{q_0\}) = \{q_0, q_1, q_2\}$$

D'où

Par exemple dans le NFA précédent



$$\delta(\{q_1, q_2\}, b) = \delta(q_1, b) \cup \delta(q_2, b) = \{q_2, q_3\} \cup \{q_2, q_3\} = \{q_2, q_3\}$$

$$\delta(\{q_1, q_2, q_3\}, a) = \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) = \{q_2, q_3\} \cup \{q_2, q_3\} \cup \{q_1, q_2, q_3\} = \{q_1, q_2, q_3\}$$

Pour le DFA, que nous voulons construire il faut donner la cinq composants:
 l'ensemble des états $Q = \mathcal{P}(Q)$ l'élément d'entrée $Z = \Sigma$
 la fonction de transition δ' l'état initial q_0 et l'ensemble des états finaux F
 prolongement de δ en $\mathcal{P}(Q) \times \Sigma$ $= \{q_0\}$ $F' = \{x \in \mathcal{P}(Q) \mid x \cap F \neq \emptyset\}$

Exemple:

$Z = \{a, b\}$ $\mathcal{P}(Q) = \{\emptyset, \{q_1\}, \{q_2\}, \{q_3\}, \{q_1, q_2\}, \{q_1, q_3\}, \{q_2, q_3\}, \{q_1, q_2, q_3\}\}$

On ne garde que les états accessibles à partir de l'état initial $\{q_0\}$

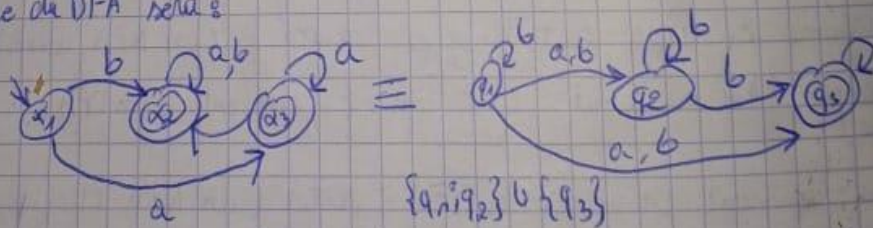
$\begin{smallmatrix} a \\ b \end{smallmatrix}$	a	b
$\alpha_1 = \{q_1, q_2\}$	$\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$
$\alpha_2 = \{q_2, q_3\}$	\emptyset	$\{q_2, q_3\}$
$\alpha_3 = \{q_1, q_2, q_3\}$	$\{q_1, q_2, q_3\}$	$\{q_2, q_3\}$

On prend alors $Q' = \{\{q_1, q_2\}, \{q_2, q_3\}, \{q_1, q_2, q_3\}\}$

ou bien $Q' = \{\alpha_1, \alpha_2, \alpha_3\}$

F' sera égale $\{\alpha_2, \alpha_3\}$

le graphe du DFA sera:



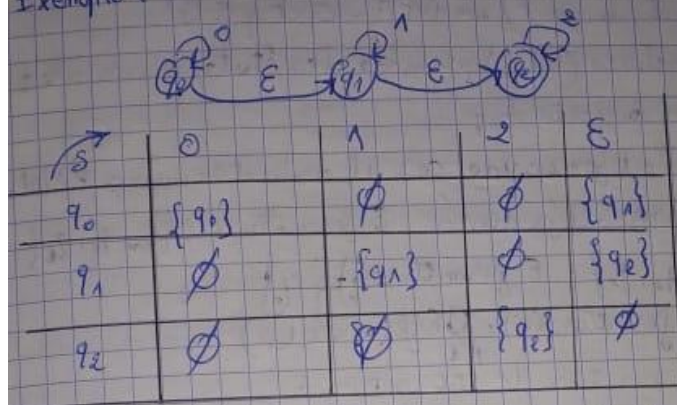
On dit que nous avons éliminé le non déterminisme
 Conclusion: le langage acceptés par les NFA sont les langages réguliers

IV - Automates finis avec des ϵ transitions

1) Définition :

c'est un automate fini qui permet des transitions entre ses états avec ϵ (c.à.d) sans consommer un symbole d'entrée)

Exemple :



Formellement Un NFA- ϵ est la donnée de 5 uplet $(Q, \Sigma, \delta, q_0, F)$

Comme un NFA sauf la fonction de transition

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$$

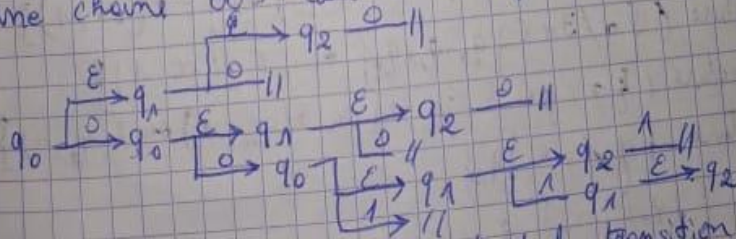
$$(q, \epsilon) \rightarrow \delta(q, \epsilon)$$

2) chaînes et langage acceptés par un NFA- ϵ

Etant donnée un NFA- $\epsilon = (Q, \Sigma, \delta, q_0, F)$ et une chaîne $w = x_1 x_2 \dots x_n \in \Sigma^+$ w sera acceptée par cet NFA- ϵ s'il existe une séquence de transitions étiquetées par $\epsilon, x_1, \dots, x_n$ et des ϵ partant de q_0 et se terminent dans un état de F

Par exemple :

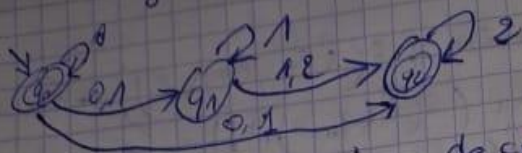
Une chaîne 001 dans le NFA- ϵ précédent



La séquence $q_0 q_0 q_1 q_1 q_2$ dont les transitions sont étiquetées respect par $00\epsilon 1\epsilon$ amène à l'état final q_2

$$\hat{\delta}(q_1, 0) = \hat{\delta}'(q_1, 0) = \varepsilon\text{-fermeture}(\delta(\hat{\delta}(q_1, \varepsilon), 0))$$

$$\varepsilon\text{-fermeture}(\emptyset)$$



Ex: Eleva le non déterminisme de cet automate

Conclusion: tout langage accepté par un NFA- ε est un langage régulier

II - Propriétés des langages réguliers

1) ils sont stables par complémentaire, par intersection et par réunion

Pour le complémentaire :

- si L est régulier alors L est accepté par un automate fini $M = (Q, \Sigma, \delta, q_0, F)$

Supposons que M est complet (si non on peut le compléter) et considérons l'auto

$M' = (Q, \Sigma, \delta, q_0, F')$ Donc toute chaîne non acceptée par M sera

acceptée par M' et vice versa Autrement dit $L(M) = \Sigma^* - L(M') = \bar{L}(M')$

Pour l'intersection : soit $L_1 = L(M_1)$ et $L_2 = L(M_2)$ deux

langages réguliers

avec $M_1 = (Q_1, \Sigma, \delta_1, q_0, F_1)$ et $M_2 = (Q_2, \Sigma, \delta_2, p_0, F_2)$

Pour garder les résultats des deux examens celui de M_1 et celui de

considérons l'ensemble $Q = Q_1 \times Q_2$

À partir de l'état initial (q_0, p_0) et qu'on suppose comme état

commençons l'examen, à l'aide de la fonction de transition δ

$$\delta: Q \times \Sigma \rightarrow Q$$

$$\delta((p, q), a) = (\delta_1(q, a), \delta_2(p, a))$$

$$w \in L_1 \cap L_2 \Rightarrow w \in L_1 \text{ et } w \in L_2$$

$$\Rightarrow \delta_1(q_0, w) \in F_1 \text{ et } \delta_2(p_0, w) \in F_2$$

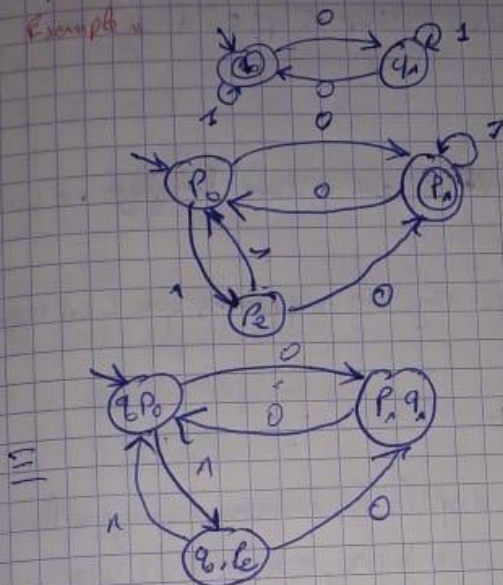
$$\Rightarrow (\delta_1(q_0, w), \delta_2(p_0, w)) \in F_1 \times F_2$$

$$\text{Posons donc } F = F_1 \times F_2$$

Donc le DFA $(Q_1 \times Q_2, \Sigma, \delta, (q_0, p_0), F_1 \cup F_2)$ accepte bien $L_1 \cap L_2$
 pour la réunion on a : $A \cup B = \overline{A \cap B}$

Ce qui implique que : $A \cup B = \overline{A \cap B}$ est pasq'on a la stabilité pour complémentaire et par intersection alors on a la stabilité par \cap

Exemple :

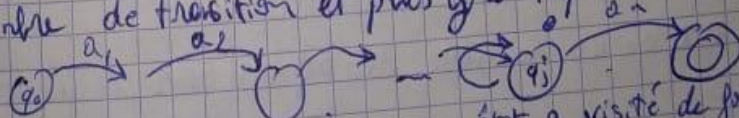


2) Lemme de gonflement ("pumping Lemma")

Si L est un langage régulier et infini alors $\exists n \in \mathbb{N}$ tel que $\forall w$
 si $|w| \geq n$ alors on peut écrire w sous la forme $w = u x v$, u, x
 et $x \neq \epsilon$ en plus $\forall n \in \mathbb{N} \quad u x^n v \in L$

En effet : Soit L infini accepté par un DFA $M = (Q, \Sigma, \delta, q_0, F)$
 considérons $n = \text{card}(Q)$ si $w = a_1 a_2 \dots a_m \in L$ avec $m \geq n$
 voyons la séquence des transitions étiquetées par les symboles, et
 de q_0 et arrivant à état $q_f \in F$

le nbre de transition et plus grand que le nombre des états



Donc il y'a au moins un état q_i visité de fois
 autrement dit il y'a une boucle dans cette séquence. notons μ
 chaîne qui nous a mené de q_0 à l'état q_i

et x la sous chaîne qui a bouclé d'une façon indirecte sur q_i et v la sous chaîne qui nous amène de q_i vers l'état final $w = uvx$.
 En plus si on redouble x plusieurs fois c'est parcourir la boucle plusieurs fois, la chaîne uvx va arriver à l'état final de même si on annule x de la chaîne w on arrive à q_i c'est uv .
 D'où vient $uv \in L$

Application:

On utilise ce lemme pour montrer que certains langages ne sont pas réguliers.

Exemple: $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas régulier supposons que L est régulier.

$$a^n b^n = uvx$$

$$\underbrace{a \dots a}_{n \text{ fois}} \underbrace{b \dots b}_{n \text{ fois}} = uvx$$

Supposons que L est régulier, d'après le lemme de gonflement $\exists n \in \mathbb{N}$ / Pour tout $w \in L$ de longueur $> n$, On peut écrire

~~$w = uvx$~~ avec $x \neq \epsilon$ et pour tout $i \in \mathbb{N}$, $u x^i v \in L$

En particulier la chaîne $w = a^n b^n$ $|w| = 2n > n$ alors $w = uvx$ donc $u \neq \epsilon$ et $u x^i v \in L \forall i \in \mathbb{N}$

$$w = \underbrace{a \dots a}_{n \text{ fois}} \underbrace{b \dots b}_{n \text{ fois}} = uvx$$

3 cas possible: x composé seulement des a
 x composé seulement des b
 x composé des a et des b

1^{er} cas: $u x^2 v$ contient un nombre de a supérieur au nombre de b
 $u x x v \Rightarrow u x^2 v \notin L$ contradiction

2^{es} cas: Donc $u x^0 v = uv$ le nombre des b < nombre des a , ce qui est cont avec $u x^0 v \in L$

3^{es} cas: $u x^3 v \in L$ et contient des a plus les b
 $w = uv = a^k a^l b^{n-l} \Rightarrow u x x v = a^k a^l a^l b^{n-l} = a^k a^{2l} b^{n-l}$
 donc L n'est pas

Ex 08. Mq: les langages suivants ne sont pas réguliers

$$\{0^i 1^j 0^{i+j} / i, j \in \mathbb{N}\}, \{a^k / k \in \mathbb{N}\}, \{p^p / p \text{ premier}\}$$

Chapitre 2 : Les expressions régulières "régulières" et press

1. Définition : soit Σ un alphabet. On définit les expressions régulières (re) et les ensembles qu'elles dénotent d'une manière récursive par :

1) \emptyset , \emptyset et s ($s \in \Sigma$) sont des re. Elles dénotent respectivement \emptyset , $\{s\}$ et $\{s\}$.

2) si r_1 et r_2 sont des expressions régulières sur Σ qui dénotent les ensembles X_1 et X_2 alors $r_1 + r_2$, $r_1 r_2$ et r_1^* sont des re qui dénotent respectivement les ensembles $X_1 \cup X_2$, $X_1 X_2$ et X_1^* .

N.B : qu'une ~~re~~ re peut contenir des parenthèses :

si r est une re qui dénote X alors (r) et aussi une re qui dénote le même ensemble X .

En plus : dans l'absence des parenthèses l'écriture de Kleene* qui est plus propriétaire de concaténation et cette dernière est plus propriétaire que

Exemples : sur $\{0, 1\}$

0.1^* dénote l'ensemble $\{0, 01, 011, 0111, \dots\}$

$(11^*)^* + (0.0)^*$ dénote $\{\epsilon, 00, 11, 0000, 1111, \dots\}$

{chaîne de longueur paire et ne se compose d'un seul symbole 0 ou 1}

$((0+1)(0+1))^*$ dénote $\{w \in \Sigma^* / |w| \text{ est paire}\}$

1.1^* et 1^*1 dénote $\{1, 11, 111, \dots\}$ {chaînes non vides des

Notation : On utilise l'opérateur κ^+ (où κ est une re) pour désigner l'ense

$X^+ = \{x\}$ où x est dénoté par κ . Autrement dit $\kappa^+ = \kappa \kappa^* = \kappa^* \kappa$

Théorème : Tout ensemble dénoté par une re est un langage rég

démonstration : (tout ensemble) par récurrence sur le nombre des opé

dans une re

Base : re contenant 0 opérateur se sont \emptyset , \emptyset et

① ②

②



~~car~~ Supposons que l'hypothèse est

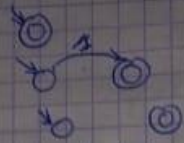
$$\begin{cases} \mu = \mu_1 + \mu_2 & \text{su} \\ \mu = \mu_1 \cdot \mu_2 & \text{di} \\ \mu = \mu_1 \cdot \mu_2 & \text{di} \end{cases}$$
 $\pi_2 \text{ ext } (n)$

avec $s \in \Sigma$ qui dénotent \emptyset accepté par l'automate \odot \odot

$\{ \epsilon \}$

$\{ s \}$

$\{ \emptyset \}$



$L(a) = \{ \epsilon \}$

$L(a) = \{ s \}$

$L(a) = \{ \emptyset \}$

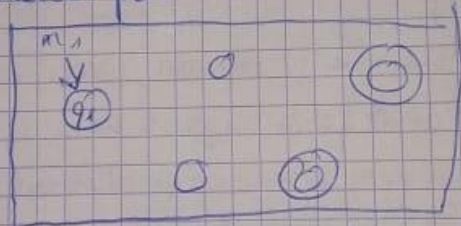
sont tous des langages réguliers. Supposons que l'hypothèse est vérifiée pour toute x contenant un nbre d'opérateur $\leq n$ et soit x contenant $(n+1)$ opérateur. Et soit x un x contenant $(n+1)$ opérateurs.

Il y a 3 cas possibles pour $x = x_1 + x_2$ ou $x = x_1 x_2$ ou $x = x_1^*$ avec nbre d'opérateur dans $x_1 \leq n$ dans $x_2 \leq n$

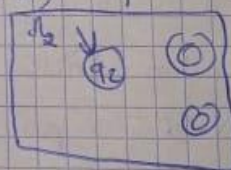
avec $\{ n \}$ d'opérateur dans x_1 et $\{ n \}$ d'opérateur dans x_2

1^{er} cas $x = x_1 + x_2$ puisque le nombre d'opérateur dans x_1 est $\leq n$ d'après l'hypothèse de récurrence $L(x_1)$ est régulier.

Alors $\exists M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ automate fini tel que $L(x_1) = L(M_1)$ peut être schématisé par :

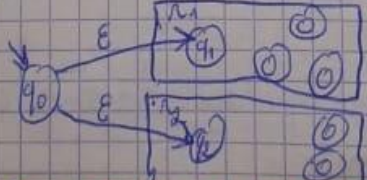


De même pour $M_2 \exists M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ tel que $L(x_2) = L(M_2)$



Noter bien $L(x) = L(x_1) \cup L(x_2)$

considérons l'automate



$(Q_1 \cup Q_2 \cup \{q_0\}, \Sigma, \delta, q_0, F_1 \cup F_2)$ avec :
 $\delta(q_0, \epsilon) = \{q_1, q_2\}$
 $\delta(q, s) = \begin{cases} \delta_1(q, s) & \text{si } q \in Q_1 \\ \delta_2(q, s) & \text{si } q \in Q_2 \end{cases}$

Il est clair que $L(M) = L(M_1) \cup L(M_2)$

Il est clair que $L(M) = L(M_1) \cup L(M_2)$

$\Rightarrow L(M) = L(M_1) \cup L(M_2)$ est un langage régulier.

2^{ème} cas : $K = K_1 \times K_2$ avec nbre d'opération dans K_1 est $\leq n$ et dans K_2 est $\leq n$

même conclusions que 1^{er} cas $\exists M_1 / L(M_1) = L(K_1)$;

$\exists M_2 / L(M_2) = L(K_2)$;

Considérons l'automate finis :



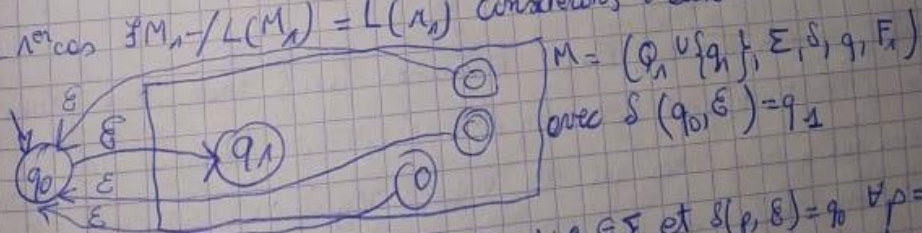
$L(M) = L(M_1) \cdot L(M_2)$

$M = (Q_1 \cup Q_2; \Sigma; \delta, q_1, F_1)$ avec $\delta(q, s) = \begin{cases} \delta_1(q, s) & \text{si } q \in Q_1 \\ \delta_2(q, s) & \text{si } q \in Q_2 \end{cases}$

et $\delta(p, \epsilon) = q_2 \quad \forall p \in F_2$

$L(M) = L(K_1) \cdot L(K_2) = L(K)$ $\Rightarrow L(K)$ est régulier

3^{ème} cas : $K = K_1^*$ est le nbre d'opération ; dans K_1 est $\leq n$, comme dans le 1^{er} cas $\exists M_1 / L(M_1) = L(K_1)$ considérons l'automate



$M = (Q_1 \cup \{q_1\}, \Sigma, \delta, q_0, F_1)$
avec $\delta(q_0, \epsilon) = q_1$

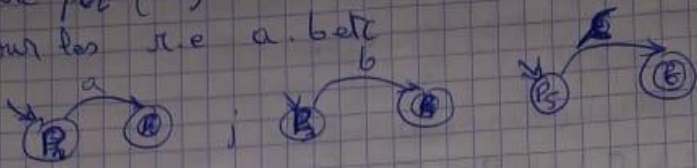
$\delta(q, s) = \delta_1(q, s) \quad \forall q \in Q_1 \quad \forall s \in \Sigma$ et $\delta(p, \epsilon) = q_0 \quad \forall p \in F_1$

on a $L(M) = (L(M_1))^* = (L(K_1))^* = L(K_1^*) \Rightarrow L(K_1^*)$ est un langage

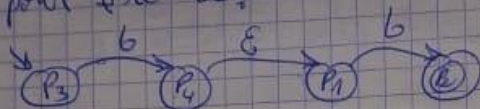
régulier

// si q_0 n'est pas un F.F. alors $L(M) = L(M_2)$

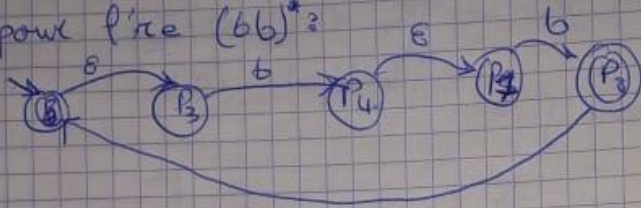
Exemple : Construire un FA qui reconnaît le langage
 dénoté par $(bb)^*a + c$ sur l'alphabet $\Sigma = \{a, b, c\}$
 Pour les a et b etc



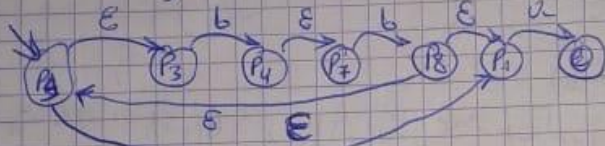
pour l'axe bb^* :



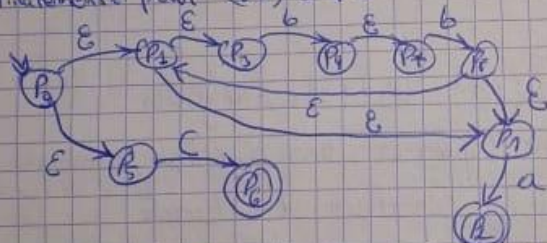
pour l'axe $(bb)^*$:



Pour $(b.b)^* . a$:



Finalement pour $(bb)^*a + c$



~~Remarque~~ On peut aussi Exo: Éliminer les ϵ transition puis le
 non déterministe pour avoir un DFA acceptant $(bb)^*a + c$

Remarque : On a prouvé que les langages réguliers sont aussi stables
 par concaténation et par l'étoile de Kleene.

- On peut aussi confirmer que L^* est un langage régulier ($\forall L$ exprès
 régulier).

Propriétés Pour tout $x, y \in \Sigma^*$ et ϵ

$$x \cdot \epsilon = \epsilon \cdot x = x$$

$$\epsilon^* = \epsilon$$

$$x \cdot \emptyset = \emptyset \cdot x = \emptyset$$

$$\emptyset^* = \epsilon$$

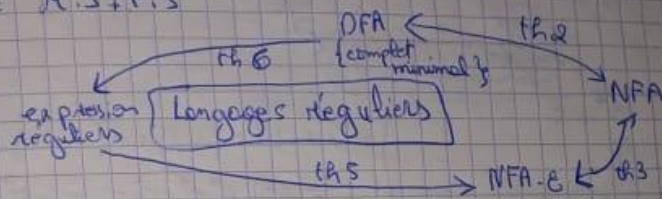
$$(x^*)^* = x^*$$

$$(x+y)^* \neq x^* + y^* \quad \text{en général}$$

$$(x^*)^* \neq x^* + y^*$$

$$x \cdot (y+z) = x \cdot y + x \cdot z$$

$$(x+y) \cdot z = x \cdot z + y \cdot z$$



Passage d'un DFA à l'expression régulière

théorème Pour DFA $M = (Q, \Sigma, \delta, q_0, F)$ il existe un $x \in \Sigma^*$

tel que $L(M) = L(x)$

dém. Numérotés les états de Q soit $Q = \{q_0, q_1, \dots, q_n\}$ Posons $R(i, j, k)$

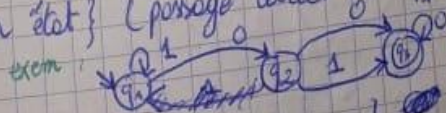
l'ensemble des chaînes de Σ^* qui amènent de l'état q_i à q_j en ne visitant que des états d'indice $\leq k$

$$L(M) = \bigcup_{q_i \in F} R(1, j, n+1)$$

Autrement dit : l'expression régulière qui dénote $L(M)$ par la somme des $x \in \Sigma^*$ qui dénotent les $R(1, j, n+1)$ (où $q_j \in F$)

Notons régulier l' $x \in \Sigma^*$ qui dénote $R(i, j, k)$

Remarquons que $R(i, j, 1) = \{w \in \Sigma^* / w \text{ amène de } q_i \text{ à } q_j \text{ sans visiter aucun état}\}$ (passage direct de q_i à q_j)



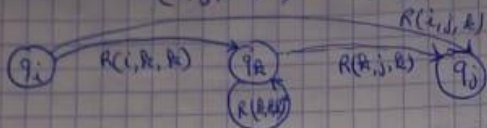
$$x(1, 1, 1) = 1^*$$

$$R(1, 1, 1) = \{\epsilon, 1, 11, 111, \dots\}$$

$$R(1, 2, 1) = \{0\} \quad r(1, 2, 1) = 0$$

$$R(1,3,1) = \emptyset \quad \mu(1,3,1) = \emptyset$$

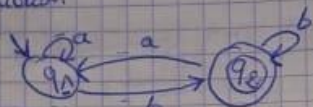
$\alpha(-1, 3, 1) = 0$ $\alpha(1, 3, 1) = 0$
 voyons comment passer de $\alpha(x, y, z)$ quelle relation existe entre

$$R(i, j, k+1) \text{ et } R(i, j, k)$$


$$\Rightarrow R(i, j, R+1) = R(i, j, R) \cup R(i, R, R) \cdot R^*(R, R, \frac{1}{k}) \cdot R(R, j, R)$$

Pour les $\kappa \in \text{onax}$ $\kappa(i, j, A+1) = \kappa(i, j, k) + \kappa(i, k, k) \cdot \kappa^*(k, k, k) \cdot \kappa(k, j, k)$

Exemple d'application



$\kappa(1,1,1)$	$a^* + \varepsilon$	$\kappa(1,1,2) = \kappa(1,1,1) + \kappa(1,1,1) \cdot \kappa^*(1,1,1) \cdot \kappa(1,1,1)$ $= (\cancel{a^*}) \cdot \cancel{a^*} \cdot \cancel{a^*} = \cancel{a^* + a^*}$
$\kappa(1,2,1)$	b	$\kappa(1,2,2) = \kappa(1,2,1) + \kappa(1,1,1) \cdot \kappa^*(1,1,1) \cdot \kappa(1,2,1)$ $= \cancel{b} + \cancel{a^*} \cdot \cancel{b} = \cancel{b + a^* b}$
$\kappa(2,2,1)$	$b^* + \varepsilon$	$\kappa(2,2,2) = \kappa(2,2,1) + \kappa(2,1,1) \cdot \kappa^*(1,1,1) \cdot \kappa(1,2,1)$ $= \cancel{b^*} + \cancel{a^*} \cdot \cancel{b} = \cancel{b^* + a^* b}$
$\kappa(2,1,1)$	a	$\kappa(2,1,2) = \kappa(2,1,1) + \kappa(2,1,1) \cdot \kappa^*(1,1,1) \cdot \kappa(1,2,1)$ $= \cancel{a} + \cancel{a^*} \cdot \cancel{b} = \cancel{a + a^* b}$
$\kappa(2,1,2)$	a^*	$\kappa(2,2,2) = \kappa(2,2,1) + \kappa(2,1,1) \cdot \kappa^*(1,1,1) \cdot \kappa(1,2,1)$ $= \cancel{b^*} + \cancel{a^*} \cdot \cancel{b} = \cancel{b^* + a^* b}$
$\kappa(1,2,2)$	$a^* b$	
$\kappa(2,1,2)$	a^*	
$\kappa(2,2,2)$	$\varepsilon + (\varepsilon + a^*) b = \varepsilon + a^* b$	

$$= (b + \varepsilon) + a \cdot (a + \varepsilon)^* \cdot b = b + \varepsilon, \quad a \cdot a^* \cdot b = b + \varepsilon, \quad a^+ b$$

$$\kappa(1, 1, 2) = \kappa(1, 1, 1) + \kappa(1, 1, 1) \cdot \kappa^*(1, 1, 1) \cdot \kappa(1, 1, 1) = (a + \varepsilon) + (a + \varepsilon)^*(a + \varepsilon)$$

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

$$\kappa(1, e, z) = b + (a + \delta)(a + \delta)^* \cdot b = \cancel{b + a + \delta} = \delta b + a^* b = (\delta + a^*) b = a^* b$$

l'exp reg équivalente à notre DFA est $\lambda(1, 2, 3)$

$$\begin{aligned} \kappa(1, 2, 3) &= \kappa(1, 2, 2) + \kappa(1, 2, 1) \kappa^*(2, 2, 2) \kappa(2, 2, 1) \\ &= a^*b + a^*b(8 + a^*b)^* (E + a^*b) = \\ &= a^*b + a^*b(a^*b)^* \quad \text{car } \alpha^*(8 + \alpha) = \alpha^* \cdot \alpha^* \end{aligned}$$

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

Chapitre 3: les grammaires hors contexte

Définition formelle:

Une grammaire hors contexte est la donnée d'un quadruplet (V, T, P, S)

où V : ensemble des symboles non terminaux

T : ensemble fini des symboles terminaux

S : symbole initial $\in V$

P : ensemble finis des productions (des règles)

Une production (ou une règle) est de la forme $X \rightarrow \alpha$

où $X \in V$ et $\alpha \in (V \cup T)^*$

Exemple:

$$G_1 = (\{Affect, Exp\}, \{id, Nbre, (,), +, *, =, ;\}, P_1, Affect)$$

où $P_1 = \left\{ \begin{array}{l} Affect \rightarrow id = Exp; \\ Exp \rightarrow id \\ Exp \rightarrow Nbre \\ Exp \rightarrow (Exp) \\ Exp \rightarrow Exp + Exp \\ Exp \rightarrow Exp * Exp \end{array} \right.$

On peut écrire $\left\{ \begin{array}{l} Affect \rightarrow id = Exp \\ Exp \rightarrow id | Nbre | (Exp) \\ Exp \rightarrow Exp + Exp | Exp * Exp \end{array} \right.$

$$G_2 = (\{Id, Alph\}, \{A, B, \dots, Z, a, b, \dots, z, \dots, 0, \dots, 9\}, P_2, Id)$$

$$P_2 = \left\{ \begin{array}{l} Id \rightarrow Alph | Alph Id | Alph 0 | Alph 1 | Alph Z | \dots | Alph 9 \\ Alph \rightarrow A | B | C | \dots | Z | \dots | a | b | \dots | z \end{array} \right.$$

$$G_3 = (\{x\}, \{a, b\}, P_3, x) \quad P_3 = \{x \rightarrow aKb | \epsilon\}$$

Dérivation et langage généré:

Etant donné une CFG (contexte Free Grammar) $G = (V, T, P, S)$ on définit deux relations de dérivation \Rightarrow (dérive) \Rightarrow^* (dérive indirecte) sur les chaînes de $(V \cup T)^*$ par:

si $A \rightarrow B$ est une production dans P

alors $\alpha \beta \gamma \Rightarrow \alpha \beta \gamma$ $\forall \alpha, \gamma \in (N \cup T)^*$

et si $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \dots \Rightarrow \alpha_n$

Alors on écrit $\alpha_1 \Rightarrow \alpha_n$

c.à.d. \Rightarrow remplace γ par une ou plusieurs \Rightarrow Par exemple

Par exemple : dans G_1 $(Exp) \Rightarrow (Exp) (Exp) \Rightarrow (Nbre + Exp)$

Dans G_2 $Id \Rightarrow Alph Id \Rightarrow Alph Alph Id \Rightarrow Alph Alph Alph Id \Rightarrow Alph S Id$

Dans G_3 $x \Rightarrow a x b \Rightarrow a a x b b \Rightarrow a a b b$

On peut aussi écrire

$(Exp) \Rightarrow^* (Nbre + Nbre + Id)$ dans G_1

$Id \Rightarrow^* Alph A X S$ dans G_2

$x \Rightarrow^* a a a a b b b b$ dans G_3

une chaîne w de T^* est dite générée (ou engendrée) par la CFG

G si et seulement si $s \Rightarrow^* w$

L'ensemble de toutes les chaînes générées par G s'appelle langage généré par G et sera noté $L(G)$ et on dit que c'est un langage hors contexte

Exemple : dans G_1 est $nbre + nbre + id$; est générée dans G_1

$A X S$ est générée dans G_2

$\epsilon, a, b, a a b b$, sont générés par la CFG G_3

Ex 1 $G_2 = \{ Id \rightarrow Alph | Alph Alph | Alph Alph | \dots | Alph Alph \}$
 $\{ Alph \rightarrow A | B | C | \dots | Z | - | a | b | \dots | z \}$

pour G_3 on peut vérifier facilement qu'elle engendre le langage

$L(G_3) = \{ \epsilon, a, b, a a b b, \dots, a^n b^n, \dots \}$

La grammaire $\begin{cases} E \rightarrow E + E | E * E | E / E | E - E \\ E \rightarrow (E) N \\ N \rightarrow NN | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 \end{cases}$

$(\{E, N\}, \{0, \dots, 9, +, -, *, /, (,)\}, E, P_4)$

La suite des dérivations par $(15 + 2 * 3 / 4)$ à partir de E

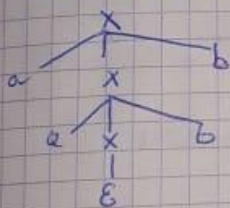
$E \Rightarrow (E) \Rightarrow (E+E) \Rightarrow (E+E+E) \Rightarrow (N+E+E) \Rightarrow (N+N+E) \Rightarrow (N+N*N)$
 $\Rightarrow (N*N+N*N) \Rightarrow (1N+N*N) \Rightarrow (15+N*N) \Rightarrow (15+2*N) \Rightarrow (15+2*N*N)$
 $\Rightarrow (15+2*3N) \Rightarrow (15+2*3*1N) \Rightarrow (15+2*3*4)$
 On peut écrire $E^* \Rightarrow (15+2*3*4)$ et déduire que $(15+2*3*4) \in L(G_4)$
Arbre de dérivation :

Soit $G = (V, T, P, S)$ une CFG pour toutes chaînes $w = s_1 s_2 \dots s_n \in L(G)$ correspond un arbre se caractériser par :

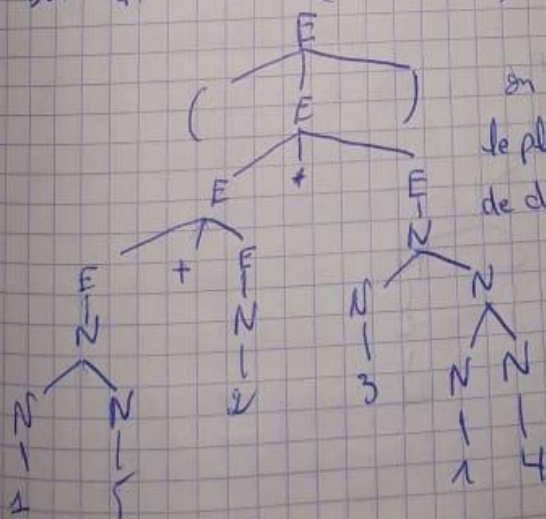
- 1) La racine est S
- 2) Les feuilles, (la gauche à droite) sont s_1, s_2, \dots, s_n
- 3) Un nœud admet A_1, A_2, \dots, A_p comme des fils
 si $x \rightarrow A_1 A_2 \dots A_p$ est une production dans P
- 4) si un nœud est ϵ alors il sera le seul fils de son père (il n'a pas de frères)

Exemple :

Dans G_2 la chaîne $aabb$ admet l'arbre de dérivation :



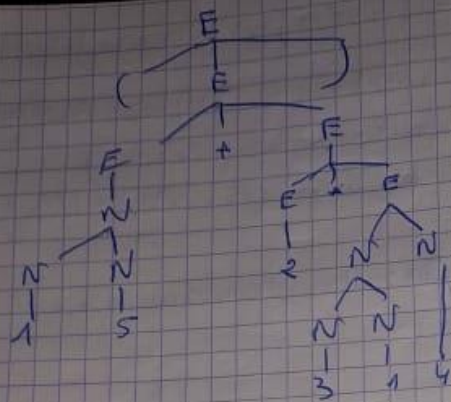
Dans G_4 la chaîne $(15+2*3*4)$ admet l'arbre



on utilise le principe "descendant le plus à gauche" pour lire un arbre de dérivation

$$15 + 2 \times 3 \times 4 = 15 + 2 \times 12$$

Ainsi d'un arbre suivant :



$$15 + (2 * 3 * 4)$$

$$15 + 6 * 4 = 643$$

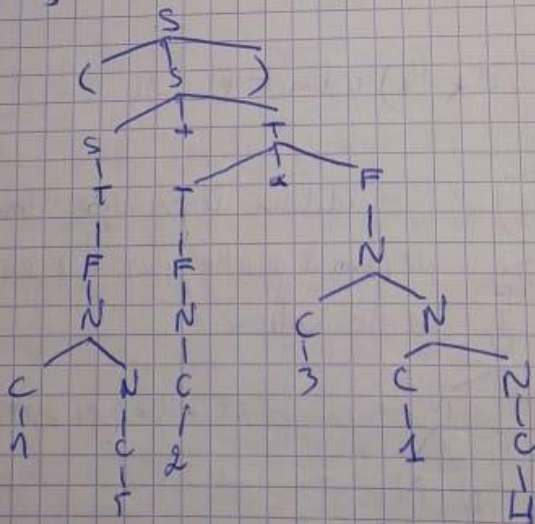
Notre chaînes à deux arbres différents

Dans tel cas, on dit que la CFG est ambiguë.
Il faut alors développer une autre CFG qui est équivalente à G_4 (génère de même langage que G_4) qui est non ambiguë.

Considérons par exemple la CFG

$$G_5 \left\{ \begin{array}{l} S \rightarrow (S) \mid S + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow N \\ N \rightarrow C \mid CN \end{array} \right. \quad C \rightarrow D \mid 1 \mid 2 \mid 3 \mid \dots \mid 9$$

Avec G_5 notre chaîne n'aura qu'un seul arbre



Grammaire régulière

Ce sont les CFG dans chaque production est de la forme
 $X \rightarrow s$ ou $X \rightarrow \epsilon$ ou $X \rightarrow sy$ avec $x, y \in V$ et $s \in T$
 Par exemple :

$$G = (\{A, B, C\}, \{0, 1\}, P, A)$$

$$P = \begin{cases} A \rightarrow 0B & | 1A \\ B \rightarrow 0C & | 1A \\ C \rightarrow 0C & | \epsilon & | 1A \end{cases}$$

G peut générer 00, 0100, 11010100

$$A \Rightarrow 0B \Rightarrow 00B \Rightarrow 00$$

$$A \Rightarrow 0B \Rightarrow 01A \Rightarrow 0100$$

Grammaire régulière

R_g : Toute grammaire régulière est équivalente à un CFG dont les productions sont de la forme $X \rightarrow sy$ ou $X \rightarrow \epsilon$ avec X, Y symbole non terminaux et s symbole terminal

en effet : si une production est de la forme $A \rightarrow s$

On peut le remplacer avec $\begin{cases} A \rightarrow sZ \\ Z \rightarrow \epsilon \end{cases}$

où Z est un nouveau symbole non terminale

Théorème :

Soient Σ un alphabet est $L \subseteq \Sigma^*$ est une langue régulière si et seulement si L est engendré par une grammaire régulière.

En effet (\Leftarrow) si $L = L(G)$ avec $G = (N, T, P, S)$ grammaire hors contexte rég considérons le DFA M dont :

- L'ensemble des états = N

- L'alphabet T

- L'état initial est S

- Les états finaux seront les non terminaux x tel que $x \rightarrow B \in T$

- La fonction de transition S sera définie $S(A, s) = B$ si $A \rightarrow$

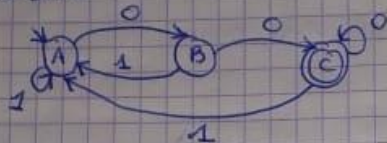
$\forall A, B \in M$ et $\forall a \in T$

Exemple :

pour la CFG :

$$\begin{cases} A \rightarrow 0B / 1A \\ B \rightarrow 0C / 1A \\ C \rightarrow 0C / 1A / \epsilon \end{cases}$$

On aura le DFA :



00 ; 0100 car $A \Rightarrow 0B \Rightarrow 01A \Rightarrow 010B \Rightarrow 0100C$

Toute chaîne générée par $L(G)$ est acceptée par l'automate M . Donc $L(G) = L(M)$
 autrement dit $L(G)$ est un langage régulier

$w \in L(G) \Rightarrow S \xRightarrow{*} w \quad w = s_1 s_2 \dots s_k$

$S \Rightarrow s_1 x \Rightarrow s_1 s_2 y \Rightarrow s_1 s_2 s_3 z \Rightarrow \dots \Rightarrow s_1 s_2 \dots s_k R \Rightarrow w$

$S \Rightarrow s_1 x \quad x \Rightarrow s_2 y \quad y \Rightarrow s_3 z \quad \dots \quad R \Rightarrow E$

$S(s_1, s_1) = x, S(x, s_2) = y \quad \dots \quad S(s_k, s_k) \in F$

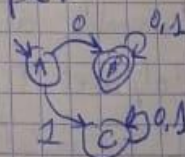
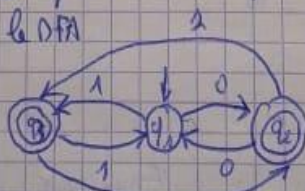
(\Rightarrow) si L est un langage régulier soit $M = (Q, \Sigma, \delta, q, F)$ un DFA qui l'accepte
 considérons le CFG G définie par

- l'ensemble des symboles non terminaux = Q
- l'ensemble des symboles terminaux = Σ
- le symbole initial est q

les productions seront construit de la manière suivante

$p \rightarrow sq$ si $S(p, s) = q$ dans M et $p \rightarrow \epsilon$ si $p \in F$

Par exemple : pour le DFA



On aura le GRC $(\{q_0, q_1, q_2\}, \{0, 1\}, P, q_0)$

$\begin{cases} A \rightarrow 0B / 1C \\ B \rightarrow 0B / 1B / \epsilon \\ C \rightarrow 0C / 1C \end{cases}$

$$L(G) = L(M)$$

Toute r.e sur Σ est équivalente à un CFG

\emptyset é equivalente a $S \rightarrow S$

$$\lambda \text{ (avec } \lambda \in \mathbb{Z}) \text{ équivaut à } \begin{cases} S \rightarrow \lambda X \\ X \rightarrow E \end{cases}$$

Considerons la CFG $G = (N, \cup_{i=1}^n U_i(S), \Sigma, P, S)$

$L(G) = L(G_1) \cup L(G_2)$ c'est le langage dénoté par $\kappa_1 + \kappa_2$

Considérons le CFG $G' = (N_1 \cup N_2 \cup \{s'\}, \Sigma, P', s')$ avec $P' = P_1 \cup P_2 \cup \{s' \rightarrow s_1 s_2\}$.

Si $K = K_n$

Si $K = K_n$
 Considérons la grammaire hors contexte $G'' = (N, \cup \{S''\}, \Sigma, P'', S'')$

on $P'' = P \cup \{S'' \rightarrow S''S_1 \mid \varepsilon\}$

[illegible]

Conséquence :
Les langages hors contexte sont stable par réunion, la concaténation et l'étoile de Kleene

Exemple $O^*(1+2)^3$

la forme normale de Chomsky (N.F.C)

Exemple 6 (NFC)
 La forme normale de Chomsky (N.F.C)
 Un CFG $G = (N, T, P, S)$ est dite en NFC si toute production dans
 est de la forme $X \rightarrow YZ$ ou $X \rightarrow s$ avec $X, Y, Z \in N$ et $s \in T$

théorème 8 : Tout langage sans contexte sans ϵ peut être généré par une grammaire en NFC

En effet :

Soit $G = (N, T, E, S)$ une CFG qui génère le langage L (avec $\epsilon \notin L$)
et soit $X \rightarrow \alpha$ une production de G

$$\alpha \neq \epsilon \text{ (car } \epsilon \notin L)$$

1^{er} cas : $|\alpha| = 1$ c-à-d $X \rightarrow y$ ou $X \rightarrow s$ ($x, y \in N, s \in T$)

$X \rightarrow s$ respecte déjà la NFC

Pour $X \rightarrow y$ on cherche la production de y

si $y \rightarrow \beta \in P$ on ajoute $X \rightarrow \beta$ dans P et on élimine $X \rightarrow y$
même cas : $|\alpha| \geq 2$

on peut écrire $X \rightarrow x_1 x_2 \dots x_k$ avec $k \geq 2$, avec $x_i \in N \cup T$

Pour x_i terminal on ajoute un symbole A_i dans N puis on ajoute

la production $A_i \rightarrow x_i$ dans P

et on remplace $X \rightarrow x_1 x_2 \dots x_k$ par $X \rightarrow x_1 A_1 x_2 A_2 \dots x_k A_k$

A la fin on a une production X

$$X \rightarrow A_1 A_2 \dots A_k \text{ avec tous les } A_i \in N$$

si $k = 2$, $X \rightarrow A_1 A_2$ respecte la NFC

si $k = 3$ $X \rightarrow A_1 A_2 A_3$ sera remplacé par

$$\begin{cases} X \rightarrow A_1 Z_1 \\ Z_1 \rightarrow A_2 A_3 \end{cases} \text{ qui respecte la NFC}$$

avec Z_1 un nouveau symbole non terminal :

Pour $k \geq 3$

$X \rightarrow A_1 A_2 \dots A_k$ sera remplacée par

$$X \rightarrow A_1 Z_1$$

$$Z_1 \rightarrow A_2 Z_2$$

$$Z_2 \rightarrow A_3 Z_3$$

$$\dots$$

$$Z_{k-2} \rightarrow A_{k-1} A_k$$

qui respecte la NFC

(Z_1, Z_2, \dots, Z_{k-1} nouveaux symboles dans N)

Exemple :

$$\begin{cases} S \rightarrow S+T / T \\ T \rightarrow T * F / F \\ F \rightarrow (s) / id \end{cases}$$

éliminer d'abord la production $T \rightarrow F$ et $S \rightarrow T$

en ajoutant les productions $T \rightarrow (s) id$ puis $S \rightarrow T * F / (s) id$

la grammaire devient :

$$\begin{cases} S \rightarrow S+T / T * F / (s) id \\ T \rightarrow T * F / (s) id \\ F \rightarrow (s) id \end{cases}$$

La production $S \rightarrow S+T$ sera remplacée dans un 1^{er} temps par

$$\begin{cases} S \rightarrow SUT \\ U \rightarrow + \end{cases}$$

ensuite par :

$$\begin{cases} S \rightarrow SR \\ R \rightarrow UT \\ U \rightarrow + \end{cases}$$

La production $S \rightarrow T * F$ sera remplacé par :

$$\begin{cases} S \rightarrow TL \\ L \rightarrow VF \\ V \rightarrow * \end{cases}$$

la production $S \rightarrow (s)$ sera remplacée par :

$$\begin{cases} S \rightarrow EM \\ E \rightarrow (\\ M \rightarrow sF \\ F \rightarrow) \end{cases}$$

Ex 4 : Mettre en NFC

$$\begin{cases} S \rightarrow aABA / aBB \\ A \rightarrow bA / b \\ B \rightarrow cB / c \end{cases}$$

$$S \rightarrow aABA \equiv \begin{cases} S \rightarrow X \cdot ABA \\ X \rightarrow a \end{cases}$$

$$\textcircled{1} \quad \begin{cases} S \rightarrow XZ \\ Z \rightarrow AB \\ E \rightarrow BA \\ X \rightarrow a \end{cases}$$

$$\textcircled{2} \quad S = aBB \equiv S \rightarrow XBB \equiv \begin{cases} S \rightarrow XF \\ F \rightarrow BB \end{cases}$$

$$\textcircled{1} \quad A \rightarrow bA \equiv \begin{cases} A \rightarrow YA \\ Y \rightarrow b \end{cases}$$

2) Mise en NFC

$$G = \begin{cases} S \rightarrow BSB \mid B \mid \epsilon \\ B \rightarrow \epsilon \end{cases}$$

le langage $L(G)$ contient ϵ .

donc on s'intéresse à la CFG de $L(G) = \{\epsilon\}$
 Éliminer $B \rightarrow \epsilon$ on obtient
 $\begin{cases} S \rightarrow BSB \mid BS \mid SB \mid \epsilon \\ B \rightarrow \epsilon \end{cases}$

~~EXOS~~

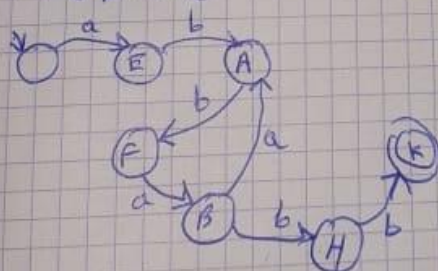
DEA

$$\begin{cases} S \rightarrow BX \mid BS \mid SB \mid AA \\ B \rightarrow AA \\ A \rightarrow \epsilon \\ X \rightarrow SB \end{cases}$$

EXS

i) :

$$\begin{cases} S \rightarrow aE \\ E \rightarrow bA \\ A \rightarrow bF \\ F \rightarrow aB \\ B \rightarrow aA \mid bH \\ H \rightarrow bK \\ K \rightarrow \epsilon \end{cases}$$



ii) CFG qui génère $L(aa^*(ab+ a)^*)$

$X \rightarrow a$ génère $L(a)$

$X \rightarrow b$ génère $L(b)$

$U \rightarrow XY$ pour ab

$V \rightarrow XY/X$ pour $ab+a$

$$\begin{cases} W \rightarrow VW/\epsilon & \text{pour } (ab+ a)^* \\ Y \rightarrow XY/X \\ Y \rightarrow a \end{cases}$$

$$x \rightarrow a x / \varepsilon$$

$$x \rightarrow a$$

$$x' \rightarrow a x' / a$$

$$R \rightarrow x' \cdot w$$

$$x' \rightarrow a x' / a$$

$$w \rightarrow v w / \varepsilon$$

$$v \rightarrow x y / x$$

$$x \rightarrow a$$

$$y \rightarrow b$$

pour a^*

pour $a a^*$

pour $a a^* (a b a)^*$

$$b \rightarrow CB \Rightarrow \begin{cases} B \rightarrow AB \\ H \rightarrow C \end{cases}$$

La NFC de notre grammaire sera :

$$\left\{ \begin{array}{l} S \rightarrow xZ / XF \\ Z \rightarrow AE \\ E \rightarrow BA \\ X \rightarrow a \end{array} \right. \quad \left\{ \begin{array}{l} F \rightarrow BB \\ A \rightarrow YA / b \\ Y \rightarrow b \\ B \rightarrow Hb / C \\ H \rightarrow C \end{array} \right.$$

$$2) \rightarrow \left\{ \begin{array}{l} S \rightarrow XY \\ X \rightarrow abb / aXb / \epsilon \\ Y \rightarrow c / cY \end{array} \right.$$

Nous devons commencer par l'élimination de la ϵ -production.

$X \rightarrow \epsilon$, la grammaire devient :

$$S \rightarrow XY / Y$$

$$X \rightarrow abb / aXb / ab$$

$$Y \rightarrow c / cY$$

Ensuite l'élimination de la production ~~non~~ unitaire $S \rightarrow Y$ on aura :

$$\left\{ \begin{array}{l} S \rightarrow XY / cY / c \\ X \rightarrow abb / aXb / ab \\ Y \rightarrow c / cY \end{array} \right.$$

$$\left\{ \begin{array}{l} S \rightarrow XY / cY / \epsilon \\ X \rightarrow AE / AF / AB \\ E \rightarrow BB \\ A \rightarrow a \\ B \rightarrow b \\ E \rightarrow XB \\ Y \rightarrow cY / c \\ C \rightarrow C \end{array} \right.$$