

# Révision examen Finale

## Alphabet et mots :

Définition: Un Alphabet " $\Sigma$ " est un ensemble fini de symboles.  
(Chiffre, Lettre, caractère spécial,...)

- Un mot de longueur  $k$  ( $|w|=k$ ) sur  $\Sigma$ :  $\Sigma^k$

Définition: Un mot est une suite finie de symbole.  
(une suite ordonnée finie)

Un mot sur un alphabet  $\Sigma$  est défini récursivement.

- Le mot vide sur  $\Sigma \in \{\epsilon\}$  est de longueur 0,  $w=\epsilon$
- $w=a \rightarrow |w|=1$  ( $w$  est un mot sur  $\Sigma^*$ )
- Si  $a=a_1 \dots a_n$  et  $b=b_1 \dots b_m$ , avec  $a$  et  $b$  des mots de taille  $n$  et  $m$  respectivement.  
Alors  $ab = a_1 \dots a_n b_1 \dots b_m$  est un mot de taille  $n+m$ . (Concaténation)
- Un mot peut aussi être défini par une longueur donné.  
 $|w| \bmod 3 \equiv 1 \rightarrow$  le mot  $w$  est de taille  $k \% 3 = 1$

Définition: Soit un alphabet  $\Sigma$  et une constante  $k \in \mathbb{N}$ ,  
On définit:

- $\Sigma^k = \{a_1 \dots a_k \mid a_i \in \Sigma, 1 \leq i \leq k\}$
- $\Sigma^* = \bigcup_{k \in \mathbb{N}} \Sigma^k$        $\Sigma^* =$  l'ensemble de tous les mots sur l'alphabet  $\Sigma$ .

Exemples:

- Alphabet: Ensemble fini de symboles.

- $\Sigma = \{a, b, c, d, \dots, x, y, z\}$  est un alphabet
- $\Sigma = \{0, 1\}$  est un alphabet (binaire)
- $\Sigma = \{\heartsuit, \diamondsuit, \clubsuit, \spadesuit\}$  est un alphabet (cartes)
- $\Sigma^* =$  l'ensemble des mots de l'alphabet  $\Sigma$   
 $\hookrightarrow \Sigma = \{0, 1\} \rightarrow \Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$
- $\Sigma^k =$  l'ensemble des mots de longueur  $k$  de  $\Sigma^*$   
 $\hookrightarrow \{w \in \Sigma^* \mid |w|=k\}$
- $\Sigma^\circ = \{\epsilon\} \cup \Sigma$

- Mot : une suite finie, possiblement vide, de symbole de  $\Sigma$ .

- $\Sigma = \{a, b, c\}$   $w = aabccabb \rightarrow w \in \Sigma^*$
- $|w| = 8 \rightarrow$  longueur de  $w$
- $|w|_a = 3 \rightarrow$  occurrence de  $a = 3$  dans  $w$ .
- $\epsilon \rightarrow$  Mot vide  $\rightarrow |\epsilon| = 0$
- $w^R \rightarrow$  l'inverse de  $w$  (Renversé)  
 $\hookrightarrow w = abc \rightarrow w^R = cba$

## Dénombrément :

Définition: Un ensemble  $X$  est dénombrable s'il existe une bijection entre  $X$  et l'ensemble  $\mathbb{N}$  d'entiers non-négatifs. (Dénombrable si on peut énumérer ses éléments)

### Preuve:

Prouvons que  $\Sigma^*$ , l'ensemble des mots sur  $\Sigma$  est dénombrable.

Rappel: pour un ensemble  $X$ , sa cardinalité (le nombre de ses éléments quand  $X$  est fini) est noté  $|X|$ .

Lemme 1: Soit  $k \in \mathbb{N}$

$$1. |\{0, 1\}^k| = 2^k$$

$$2. \sum_{i=0}^k 2^i = 2^{k+1} - 1.$$

→ Démonstration:

- Le nombre de mots de longueur  $K$  sur l'alphabet  $\{0,1\}$  est  $2^K$ , car nous avons 2 choix pour le premier symbole d'un mot, 2 choix pour le second, ..., 2 choix pour le  $K$ ème symbole.

$$\underbrace{2 \cdot 2 \cdot 2 \cdots 2}_K = 2^K \text{ possibilités.}$$

- La deuxième identité est évidemment vrai pour  $K=0$ .

$$2^0 = 2^{0+1} - 1 = 2^1 - 1 = 2 - 1 = 1. \quad \checkmark$$

Si cette identité est vérifier pour  $K$ , on a

$$\sum_{i=0}^K 2^i = 2^{K+1} - 1.$$

Si on ajoute un terme à la somme, on

Obtient :

$$\sum_{i=0}^{K+1} 2^i = 2^{K+1} - 1 + 2^{K+1} = 2 \cdot 2^{K+1} - 1 = 2^{K+2} - 1$$

Ainsi la récurrence de l'identité est vrai  $\forall K$ . On peut également voir la deuxième identité comme le résultat de la substitution de 2 pour  $x$  dans l'identité évidente

$$x^{K+1} - 1 = (x-1) \sum_{i=0}^K x^i$$

✓

**Lemme 2:** L'ensemble  $\{0,1\}^*$  est dénombrable.

→ Démonstration:

Rappelons que  $\{0,1\}^* = \bigcup_{K \in \mathbb{N}} \Sigma^K$ . Suivant cette définition, nous allons donner une bijection explicite entre  $\mathbb{N}$  et  $\{0,1\}^*$ .

Il y a  $2^K$  mots dans  $\{0,1\}^K$ . Donc pour chaque  $K$ , il y a

$$\sum_{i=0}^K 2^i = 2^{K+1} - 1 \text{ mots de longueur au plus } K.$$

On va définir la bijection  $c: \{0,1\}^* \rightarrow \mathbb{N}$  par

$$c(a_1 a_2 \dots a_K) = 2^{K+1} - 1 + n(a_1 \dots a_K),$$

où  $n(a_1 \dots a_k)$  est l'entier dont la représentation en binaire est  $a_1 \dots a_k$ , on doit également définir  $n(\varepsilon) = 0$ .

Pour prouver que l'on a vraiment une bijection, définissons l'inverse.

Soit  $c'(n) = (n+1 - 2^{\lfloor \log n + 1 \rfloor})_{\lfloor \log n + 1 \rfloor}$ , ici  $n_k$  est la représentation en binaire de  $n$  sur  $k$  bits.  
Donc si  $n=0$ , nous obtenons la représentation de  $0$  en zéro bits, soit  $\varepsilon$ .

On voit que  $c$  est surjective. Elle est également injective car si

$$c(a_1 a_2 \dots a_k) = c(a'_1 a'_2 \dots a'_m),$$

alors  $k=m$  et  $a_i = a'_i$ , parce que si, sans perdre de généralité,  $k < m$  alors

$$c(a_1 a_2 \dots a_k) = 2^{k+1} - 1 + n(a_1 a_2 \dots a_k) < 2^k - 1 + 2^k < 2^{k+1} - 1 \leq 2^m - 1 \leq c(a'_1 a'_2 \dots a'_m)$$

et quand  $k=m$  alors

$$n(a'_1 a'_2 \dots a'_m) = n(a_1 a_2 \dots a_k)$$



Théorème :

L'ensemble  $\Sigma^*$  est dénombrable  $\forall$  alphabet  $\Sigma$ .

Théorème : Soit  $X$  un ensemble, alors  $|2^X| \geq |X|$

→ Démonstration:

Puisque  $\{x\} \in 2^X$  pour tout  $x \in X$ ,  $|2^X| \geq |X|$ .  
Il faut donc prouver que  $|2^X| \neq |X|$ .

Supposons le contraire, c'est à dire  
supposons qu'il existe une bijection  $f: X \rightarrow 2^X$ .

Soit  $D = \{x \in X \mid x \notin f(x)\}$ . Puisque  $f$  est une bijection, il existe un  $d \in X$  tq  $f(d) = D$ . Mais alors  $d \in D$  ssi  $d \notin f(d)$  ssi  $d \notin D$ .

Cette impossibilité implique que  $f$  ne peut pas exister.



## Langage :

Définition : Un langage sur  $\Sigma$  est un sous-ensemble de  $\Sigma^*$ .  $L \subseteq \Sigma^*$

- On note  $\bar{L}$  le complément de  $L$   

$$\bar{L} = \Sigma^* \setminus L$$
 sauf

$$\begin{array}{l} \overline{\Sigma^*} = \emptyset \\ \overline{\emptyset} = \Sigma^* \end{array}$$

- La concaténation de  $L_1$  et  $L_2$  est  
 $L_1 \cdot L_2 = \{uv \mid u \in L_1, v \in L_2\} \neq L_2 \cdot L_1$
- La  $k$ -ième puissance de  $L$  est la concaténation de  $L$  avec lui-même  $k$  fois.

$$L^k = \underbrace{L \cdot L \cdot \dots \cdot L}_k$$

$$L^\circ = \{\epsilon\} \neq \emptyset$$

$$L' = L^\circ \cdot L = L$$

- Fermeture de Kleene (Kleene closure)  
 Soit la concaténation d'un langage avec lui-même à l'infini pour  $\forall L$  sauf  $\emptyset$  et  $L^\circ$ .

$$L^* = \{w_1 w_2 \dots w_n \mid w_i \in L, n \in \mathbb{N}\}$$

- Union de deux langages crée un langage formé de ces langages.

$$L_1 = \{aa, b\}$$

$$L_2 = \{a, bb\}$$

$$L_1 \cup L_2 = \{\epsilon, a, b, aa, bb\}$$

$$L \cup \Sigma^* = \Sigma^*$$

$$L \cup \emptyset = L$$

- Intersection de deux langages donne le

langage de leur ressemblance.

$$L_1 = \{a, b, c\}$$

$$L_2 = \{ba, b, bc\}$$

$$L_1 \cap L_2 = \{b\}$$

$$L \cap \Sigma^* = L$$

$$L \cap \emptyset = \emptyset$$

• Produit Cartésien de deux langages

$$L_1 \times L_2 = \{(w_1, w_2) \mid w_1 \in L_1 \text{ et } w_2 \in L_2\}$$

$$\Delta abb \neq (a, bb)$$

$$L_1 = \{\epsilon, a\}$$

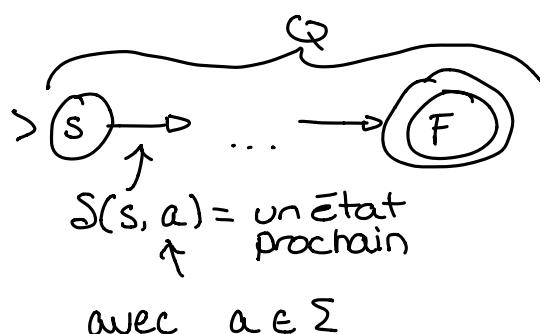
$$L_2 = \{a, b\}$$

$$L_1 \times L_2 = \left\{ \underset{L_1}{(\epsilon, a)}, \underset{L_1}{(\epsilon, b)}, \underset{L_2}{(a, a)}, \underset{L_2}{(a, b)} \right\}$$

## AFD :

Définition : Un automate fini déterministe  $M = (Q, \Sigma, S, s, F)$  est défini par :

- $Q$  : l'ensemble fini des états de  $M$
- $\Sigma$  : Un Alphabet (ensemble fini) de symbole
- $S$  :  $Q \times \Sigma \rightarrow Q$  la fonction de transition
- $s$  :  $s \in Q$  l'état initial de  $M$
- $F$  :  $F \subseteq Q$  un ensemble fini d'état acceptant.



Soit  $M = (Q, \Sigma, S, s, F)$  un AFD, et soit  $w \in \Sigma^*$ ,  
 $w = a_1 \dots a_n$ ,  $a_i \in \Sigma$  pour  $i = 1, \dots, n$ ,  $n \in \mathbb{N}$ .

Un calcul, ou exécution, de  $M$  sur  $w$  est une suite  $\{r_i\}_{i=0}^k$  ( $k=|w|$ ) d'état tel que

1.  $r_0 = s$
2.  $r_i = \delta(r_{i-1}, a_i)$  pour  $i=1, \dots, k$  et  $a_i \in \Sigma$ .

Définition : Soit  $M = (Q, \Sigma, \delta, s, F)$ , un AFD. Soit  $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$  l'extension de  $\delta$  au mot sur  $\Sigma$ , définie par :

1.  $\hat{\delta}(q, \epsilon) = q \quad \forall q \in Q;$
2.  $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a), \quad \forall w \in \Sigma^*, a \in \Sigma.$

$\hat{\delta}(q, w)$  est l'état dans lequel se trouve  $M$  après avoir lu  $w$ , en commençant dans l'état  $q$ .

On voit que  $\{r_i\}_{i=0}^k$  est un calcul de  $M$  sur  $w = a_1 \dots a_k$  si  $r_0 = s$  et  $\forall i, 1 \leq i \leq k, \hat{\delta}(s, a_1 \dots a_i) = r_i$ .

$\delta \rightarrow$  une fonction de transition  
(état présent, symbole  $\epsilon \in \Sigma$ ) = état future

$\hat{\delta} \rightarrow$   $\forall$  transitions de  $M$  de l'état de départ à un état  $r_i$ , après la lecture d'un mot  $w \in \Sigma^*$ .

(état initial, mot  $\epsilon \in \Sigma^*$ ) = état "présent"

Définition : Le langage reconnu (décidé) par  $M = (Q, \Sigma, \delta, s, F)$  un AFD est :

$$L(M) = \{w \in \Sigma^* \mid \hat{\delta}(s, w) \in F\}$$

élément de car  $F$  est un ensemble d'état acceptant donc il peut en avoir + que 1.

Un langage  $L$  est régulier ( $f_R$ ) si il existe un AFD  $M$  tel que  $L(M) = L$ .

Définition: Soit  $Z$  AFD,  $M_1$  et  $M_2$ . Ils sont équivalents ssi  $L(M_1) = L(M_2)$

$$L_1 \cup L_2 = L \rightarrow L \in \mathcal{L}_R$$

Lemme: Soit  $L_1$  et  $L_2$  deux langages régulier. Alors  $L = L_1 \cup L_2 \in \mathcal{L}_R$ .

Démonstration:

Soit  $M_i = (Q_i, \Sigma_i, S_i, s_i, F_i)$  un AFD qui reconnaît  $L_i$ , avec  $i=1,2$ .

Construisons un AFD  $M = (Q, \Sigma, S, s, F)$  qui reconnaîtra  $L$ .

Tout d'abord, on suppose que  $\Sigma = \Sigma_1 = \Sigma_2$ .

Ensuite, mettons  $Q = Q_1 \times Q_2$ ,  $S = (S_1, S_2)$  et  $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$ .

On définit  $S((q_1, q_2), a) = (S_1(q_1, a), S_2(q_2, a))$ .

Nous vérifions ensuite que  $\hat{\delta}(s_1, w) \in F$  ssi soit  $\delta(s_1, w) \in F_1$ , soit  $\delta(s_2, w) \in F_2$ .

En effet,

$\hat{\delta}(s, w) = (\hat{\delta}_1(s_1, w), \hat{\delta}_2(s_2, w)) \in F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$   
ssi soit  $\hat{\delta}_1(s_1, w) \in F_1$ , soit  $\hat{\delta}_2(s_2, w) \in F_2$ .

Pour montrer que cela fonctionne, on prétant que  $L(M) = L_1 \cup L_2$ . On prouve que  $\forall w \in \Sigma^*$   $\hat{\delta}((s_1, s_2), w) \in F$  ssi  $w \in L_1 \cup L_2$ .

$w \in L_1 \cup L_2$  ssi  $w \in L_1$  ou  $w \in L_2$  ssi  $\hat{\delta}_1(s_1, w) \in F_1$ , ou  $\hat{\delta}_2(s_2, w) \in F_2$  ssi  $\hat{\delta}((s_1, s_2), w) \in F$ .

( $\hat{\delta}_1(s_1, w) \in F_1$  et  $\hat{\delta}_2(s_2, w) \in F_2$ ) ou ( $\hat{\delta}_1(s_1, w) \in F_1$  et  $\hat{\delta}_2(s_2, w) \in F_2$ ) ssi  $\hat{\delta}((s_1, s_2), w) \in F$ .

$$L_1 \cap L_2 = L \rightarrow L \in \mathcal{L}_R$$

La preuve est la même sauf que  $F = F_1 \times F_2$

- $F_0 = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

- $F_0 = F_1 \times F_2$

Preuve que l'union de  $k$  langages régulier est régulier.

Devoir 2 Question 1 → Feb 1<sup>st</sup>

Soit  $k \in \mathbb{N}^{>1}$ . Trouver que si  $L_0, L_1, \dots, L_{k-1}$  sont des langages régulier sur le même alphabet  $\Sigma$ , alors  $\bigcup_{i=0}^{k-1} L_i$  est régulier.

Preuve par récurrence:

• Cas de base:  $k=1$

$\bigcup_{i=0}^k L_i = L_0$  est régulier  
par Hypothèse

Hypothèse de récurrence:

Pour  $n \geq 2$ , supposons que l'énoncé est vrai pour

$$\Rightarrow \bigcup_{i=0}^{n-1} L_i \text{ est régulier}$$

Pas de récurrence: Pour  $n \geq 2$  Montrer l'énoncé

$$\bigcup_{i=0}^{n-1} L_i = \left( \bigcup_{i=0}^{n-2} L_i \right) \cup L_n \text{ est régulier car:}$$

①  $\bigcup_{i=0}^{n-2} L_i$  est régulier par l'hypothèse de récurrence

②  $L_n$  est régulier par l'hypothèse du problème

③ L'union de 2 langages régulier est régulier  $\square$

Lemme 5: Soit  $L_1$  et  $L_2$ , deux langage réguliers.  
Alors il existe

tel que  $M_i = (Q_i, \Sigma_i, \delta_i, S_i, F_i)$ ,  $i=1, 2$ ,

$$L(M_i) = L_i, \text{ et } Q_1 \cap Q_2 = \emptyset$$

ensemble vide

→ Démonstration:

Puisque les 2 langages sont régulier, il existe des AFD

$M'_i = (Q'_i, \Sigma'_i, \delta'_i, s'_i, F'_i)$ ,  $i = 1, 2$ ,  
 Tel que  $L(M'_i) = L_i$ .

On définit  $M_i$  en posant

- $Q_i = Q'_i \times \Sigma'_i$
- $s_i = (s'_i, i)$
- $F_i = F'_i \times \Sigma'_i$
- $\delta_i((q, i), a) = (\delta'_i(q_i, a), i)$ .

Il est facile de voir (par récurrence) que  $\delta_i(q, w) = (\delta'_i(q, w), i)$  pour tout  $w \in \Sigma^*$  et donc que  $\delta(s_i, w) = (\delta'_i(s_i, w), i) \in (F'_i \times \Sigma'_i) = F_i$  si  $\delta'_i(s'_i, w) \in F'_i$ .

Il est évident que  $Q_1 \cap Q_2 = \emptyset$

## AFN:

Définition: Un automate Fini Non-déterministe, AFN,  $M = (Q, \Sigma, \delta, s, F)$  est défini par :

- $Q$ : un ensemble fini d'états.
- $\Sigma$ : un alphabet (ensemble fini) de symbole.
- $\delta$ :  $Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$ , une fonction de transition. ( $2^Q \rightarrow$  ensemble des parties de  $Q$ ).
- $s$ : un état initial ( $s \in Q$ ).
- $F$ : un ensemble d'état acceptant ( $F \subseteq Q$ ).

Soit  $M = (Q, \Sigma, \delta, s, F)$ , un AFN. On définit, pour tout  $q \in Q$  et tout  $k \in \mathbb{N}$ , l'ensemble  $E_k(q)$  d'état accessible à partir de  $q$  par des transitions spontanées en  $k$  étapes et l'ensemble  $E(q)$  d'état accessible par des transitions spontanées à partir de  $q$ :

1.  $E_0(q) = q$

2.  $E_{k+1} = E_k(q) \cup \left( \bigcup_{p \in E_k(q)} \delta(p, \epsilon) \right)$ , pour  $k \in \mathbb{N}$

3.  $E(q) = \bigcup_{k=0}^{\infty} E_k(q) = \bigcup_{k=0}^{\infty} E_k(q) = \bigcup_{k \in \mathbb{N}} E_k(q)$

$$= \{ p \in Q \mid \exists k \in \mathbb{N}, \exists p_0, \dots, p_k \in Q \text{ tel que } p_0 = q, p_k = p \text{ et } p_{i+1} \in \delta(p_i, \varepsilon), \text{ pour } i=0, \dots, k-1 \}.$$

Les bornes viennent du fait que si aucun nouvel état n'est ajouté à  $E_k(q)$ , alors  $E_k(q) = E_{k+1}(q)$ , pour tout  $k \geq 0$ , et si si on ajoute les état un à un, cela prend  $|Q|-1$  étapes pour épuiser  $Q$ .

Avec ceci, on peut définir, pour  $X \subseteq Q$ , l'ensemble d'états accessible par des transitions spontanées à partir des états de l'ensemble  $X$ :  $E(X) = \bigcup_{q \in X} E(q)$ .

Notons que pour tout  $X \subseteq Q$ ,  $E(E(X)) = E(X)$ .

Définition: Pour tout AFD  $M = (Q, \Sigma, \delta, s, F)$ , il existe un AFN  $M' = (Q', \Sigma', \delta', s', F')$ , équivalent.

→ Démonstration:

On définit  $M'$  en mettant:

- $Q' = Q$ ,
- $\Sigma' = \Sigma$ ,
- $s' = s$ ,
- $F' = F$ ,
- $\delta'(q, a) = \{\delta(q, a)\}$ .

On a alors que  $\hat{\delta}'(q, \varepsilon) = \{q\} = E(q)$  pour tout  $q \in Q$ , car il n'y a pas de transition spontanées.

Ceci permet de montrer (par récurrence) que  $\hat{\delta}'(q, w) = \{\hat{\delta}(q, w)\}$ . En effet pour tout  $w = ua$ ,

$$\begin{aligned} \hat{\delta}'(q, ua) &= \bigcup_{p \in \hat{\delta}'(q, u)} \bigcup_{r \in \delta'(p, a)} E(r) = \bigcup_{p \in \hat{\delta}'(q, u)} \bigcup_{r \in \delta'(p, a)} \{r\} \\ &= \bigcup_{p \in \hat{\delta}'(q, u)} \{\delta(p, a)\} = \bigcup_{p \in \{\hat{\delta}(q, u)\}} \{\delta(p, a)\} = \{\delta(\hat{\delta}(q, u), a)\} \end{aligned}$$

(Le passage de  $p \in \hat{\delta}(q, u)$  à  $p \in \{\hat{\delta}(q, u)\}$  est par hypothèse de récurrence).

On conclut que  $\hat{\delta}(s, w) \in F$ ssi  $\hat{\delta}'(s, w) \cap F' \neq \emptyset$  et donc  $w \in L(M)$ ssi  $w \in L(M')$ .

Définition: Pour tout AFN  $M = (Q, \Sigma, \delta, s, F)$ , il existe un AFD  $M' = (Q', \Sigma, \delta', s', F')$  équivalent.

→ Démonstration:

On définit  $M'$  par:

- $Q' = \{E(X) : X \in 2^Q\}$  ( $2^Q$  est l'ensemble des parties de  $Q$ )
- $\Sigma' = \Sigma$
- $S' = E(\{s\})$
- $F' = \{X \in Q' \mid X \cap F \neq \emptyset\}$
- $\delta' = Q' \times \Sigma \rightarrow Q'$

$$\delta'(X, a) = \bigcup_{q \in X} \hat{\delta}(q, a).$$

Avec ceci, on a par définition,  $\hat{\delta}'(X, \varepsilon) = X = E(X)$  et  $\hat{\delta}'(X, ua) = \hat{\delta}'(\hat{\delta}'(X, u), a)$  et on prétend que  $\hat{\delta}'(X, \omega) = \bigcup_{q \in X} \hat{\delta}(q, \omega)$ . On prouve par récurrence.

- Si  $\omega = \varepsilon$ ,  $\hat{\delta}'(X, \varepsilon) = E(X) = \bigcup_{q \in X} E(q) = \bigcup_{q \in X} \hat{\delta}(q, \varepsilon)$ ;
- Si  $\omega = ua$ ,  $u, a \in \Sigma^*$  et  $\hat{\delta}'(X, u) = \bigcup_{q \in X} \hat{\delta}(q, u)$ ,

$$\text{alors } \hat{\delta}'(X, ua) = \delta'(\hat{\delta}(X, u), a)$$

$$\begin{aligned} &= \delta'\left(\bigcup_{q \in X} \hat{\delta}'(q, u), a\right) = \bigcup_{q \in X} \bigcup_{p \in \hat{\delta}(q, u)} \hat{\delta}(p, a) \\ &= \bigcup_{q \in X} \bigcup_{p \in \hat{\delta}(q, u)} \bigcup_{r \in \delta(p, a)} E(r) \\ &= \bigcup_{q \in X} \hat{\delta}(q, ua) = \bigcup_{q \in X} \hat{\delta}(q, \omega) \end{aligned}$$

En particulier,  $\hat{\delta}'(s', \omega) = \hat{\delta}(s, \omega)$  et donc  $\hat{\delta}'(s', \omega) \in F'$  si  $\hat{\delta}(s, \omega) \cap F \neq \emptyset$  par définition de  $F$ .



## Langage Régulier:

Comme mentionné plus haut, un langage est régulier si un AFD (ou AFN) le décide.

Voir Équivalence AFN-AFD

De plus, un langage est régulier s'il peut-être défini par une expression régulière.

## Définition Formel :

L'ensemble des langage régulier sur un alphabet  $\Sigma$  est défini récursivement comme suit:

- Le langage vide  $\emptyset \in L_R$
- Pour tout symbole  $a$ ,  $a \in \Sigma$ , le langage  $L = \{a\} \in L_R$ .
- Pour un langage régulier  $A$ ,  $A^*$  (Kleene Star)  $\in L_R$ .  
De cette propriété, le langage du mot vide  $L = \{\epsilon\} \in L_R$
- Si  $A$  et  $B$  sont des langages réguliers, alors  $A \cup B$  et  $A \cdot B$  sont régulier.
- Aucun autre langage sur  $\Sigma$  est régulier.

## Exemple:

- Tout langage fini est régulier, en particulier le langage du mot vide  $\{\epsilon\} = \{\epsilon\}^* = \emptyset^*$
- Un langage non-régulier ( $L \notin L_R$ ) ex:  
 $L = \{a^n b^n \mid n \geq 0\}$  n'est pas régulier (preuve par le théorème du pompiste pour les langages réguliers). Intuitivement, le langage ne peut pas être reconnue par un Automate fini, car un automate fini ne peut pas se rappeler du nombre de  $a$  qu'il a lu avant de lire les  $b$ . Ainsi, il ne peut pas accepter le langage qui contient  $n a$  et  $n b$  pour  $n \in \mathbb{N}$ .

## - Propriété d'un langage régulier:

Un langage régulier est:

- le langage d'une expression régulière.
- accepté par un AFD.
- accepté par un AFN.
- générable par une grammaire régulière.
- accepté par une Machine de Turing
- consiste d'un nombre fini de classe d'équivalence  $R_L$ . (Myhill-Nerode)

## - Propriété de fermeture :

Un langage est régulier s'il est obtenue à partir de 2 langages régulier par:

- Union ( $L_1 \cup L_2$ )
- intersection ( $L_1 \cap L_2$ )

- concatenation ( $L_1 \cdot L_2$ )
- complement ( $\complement L$ ) (un seul langage)
- le complément relatif ( $L_1 - L_2$ )
- L'étoile de Kleene ( $L^*$ )
- La renversé ( $L^\dagger$ ) (un seul langage)

$\forall$  langages régulier est un langage hors-contexte, mais pas tout les langage hors-contexte sont régulier.

## Théorème du Pompiste (langage Régulier) :

Soit  $L$ , un langage régulier sur un alphabet  $\Sigma$ .  
Alors il existe une constante  $p \in \mathbb{N}^{>0}$  (longueur de pompage telle que  $\forall w \in L$ ,  $|w| \geq p$ ,  $\exists x, y, z \in \Sigma^*$  vérifiant :

- ①  $w = xyz$
- ②  $|xy| \leq p$
- ③  $|y| > 0$
- ④  $xy^i z \in L$ ,  $\forall i \geq 0$  (i est un entier)

"Ce théorème prouve seulement si un langage n'est pas régulier."

On fait la contraposé pour montré qu'un langage n'est pas régulier.

- On suppose que notre langage est régulier
- On choisit  $w \in L$ ,  $|w| \geq p$ , avec  $p \in \mathbb{N}$
- On prouve :

- $\forall x, y, z \in \Sigma$  tq ①  $w = xyz$  vérifiant
- ②  $|xy| \leq p$
  - ③  $|y| > 0$
  - ④  $\exists i \in \mathbb{N}$  tq  $xy^i z \notin L$

⚠️ Attention dans de rare cas, un langage non régulier peut respecté le lemme du Pompiste.

$$\text{Ex: } L = \{abc^j \mid j \geq 0\} \cup \{a^i b^j c^k \mid i, j, k \geq 0, i \neq 1\}$$

Pour montré qu'il n'est pas régulier, on utilise  $L(ab^*c^*)$  qui est régulier (Kleene star) et on utilise l'intersection.

$$L \cap L(ab^*c^*)$$

## Expression Régulière:

Définition: ER est une expression régulière sur l'alphabet  $\Sigma$  ssi (définition inductive):

1.  $ER = \emptyset$
2.  $ER = a$ , avec  $a \in \Sigma$  (comprend  $\epsilon$ )

ou encore, soit  $ER_1$  et  $ER_2$ , des expressions régulières :

1.  $ER = (ER_1) \cup (ER_2)$  ( $\cup$  est noté + dans une ER)
2.  $ER = (ER_1) \cdot (ER_2)$
3.  $ER = (ER_1)^*$

Donc :

- $\emptyset$  est une ER
- $\epsilon$  est une ER
- $a$  est une ER,  $\forall a \in \Sigma$
- $(ER_1) \cdot (ER_2)$  est une ER
- $(ER_1) + (ER_2)$  est une ER
- $(ER_1)^*$  est une ER

Définition: On définit le langage d'une expression régulière  $L(ER)$  comme suit

- $L(\emptyset) = \emptyset$
- $L(a) = \{a\}$ , pour  $a \in \Sigma$
- $L((ER_1) + (ER_2)) = L(ER_1) \cup L(ER_2)$
- $L((ER_1) \cdot (ER_2)) = L(ER_1) \cdot L(ER_2)$
- $L((ER)^*) = L(ER)^*$

Exemples:

ER =  $(a \cdot (a+b)^*) + ((a+b)^* \cdot a)$   
Le langage  $L(ER)$  décrit par ER est

$$L = \{w \mid w = a \cdot x \text{ ou } w = x \cdot a \text{ avec } x \in \Sigma^*\}$$

$L(ER)$  est l'ensemble des mots commençant ou finissant par  $a$ .

ER =  $((a+b) \cdot (a+b))^*$

$$L(ER) = \{w \mid |w| = 2k, \text{ pour } k \in \mathbb{N}\}$$

- Syntaxe des expression régulières:

- (...) : une expression régulière contient des parenthèse autour de chaque set de string.

ex:  $((a+b) \cdot (a+b))^*$

- + : représente union (ou)  
 $(a+b) = a \text{ ou } b$
- . : Concaténation
- \* : une répétition du string qui peut aller de 0 ( $\epsilon$ ) à  $\infty$

⚠ Attention:

$$01^* \Rightarrow \{0\} \cdot \{1\}^* = 0 \text{ ou } 01 \text{ ou } 011 \dots$$

Exemple:

- $0^*$ : ensemble des mots composés uniquement de 0 allant du mot vide ( $\epsilon$ ) à 0000...0...0...
- $0000^*$ : ensemble des mots qui contiennent un minimum de quatre 0.
- $(\epsilon+1)(01)^*(\epsilon+0)$ : l'ensemble de tout les mots binaires qui ne contiennent pas de 00 ou 11 en sous-string.
- $1^*(01^*01^*)^*$ : l'ensemble des mots binaire tel que le nombre de 0 est paire.

Definition: 2 expressions régulières sont équivalentes ssi elles décrivent le même langage.

$$(0+1)^* = (1+0)^* \text{ car l'union est commutatif.}$$

Définition: Soit  $L$  un langage.  $L$  est régulier ssi  $L = L(ER)$  pour une expression régulière  $ER$ .

Preuve:

Montrons que  $L = L(ER)$  pour une expression régulière  $ER$ ,  $L \in I_R$

Par induction structuré sur la définition d'une expression régulière.

- Cas de Base: les 2 cas non récursif de la définition d'expression régulière sont:
  - $ER = \emptyset$ , alors  $L(ER) \in I_R$
  - $ER = a$ , pour  $a \in \Sigma_\epsilon$ , alors  $L(a) = \{a\} \in I_R$

- Pas de récurrence:

Soit  $ER_1$  et  $ER_2$ , deux expressions régulières dont on suppose  $L(ER_1), L(ER_2) \in I_R$ .

Trois règles de construction sont à examiner:

- $ER = ER_1 \cup ER_2 : L(ER_1) \cup L(ER_2)$   
par définition de fermeture sur l'union des langages réguliers,  $L(ER_1) \cup L(ER_2) \in I_R$
- $ER = ER_1 \cdot ER_2 : L(ER_1) \cdot L(ER_2) \in I_R$
- $ER = ER_1^* : L(ER_1) \in I_R$

Par récurrence,  $L = L(ER) \in I_R$ . ✓

# Myhill-Nerode

Théorème: Soit  $\Sigma$  un alphabet et  $L \subseteq \Sigma^*$ .  
Alors les énoncés suivants sont équivalents:

- ①.  $L$  est un langage régulier
- ②.  $L$  est la réunion de certaines classes d'équivalence d'une relation d'équivalence sur  $\Sigma^*$  invariable à droite et d'index fini.
- ③. La relation  $R_L$  est d'index fini.

Démonstration:

Prouvons  $(1) \Rightarrow (2) \Rightarrow (3) \Rightarrow (1)$

- $(1) \Rightarrow (2)$ : On sait que pour  $M$  un AFD tq  $L(M) = L$ , alors la relation  $R_M$  est invariable à droite, d'index fini, et on a que

$$L = \bigcup_{x \in L} [x] = \bigcup_{q \in F} C_q.$$

- $(2) \Rightarrow (3)$ : Prouvons que toute relation  $E$  vérifiant (2) raffine  $R_L$ .

Puisque le nombre de classe d'équivalence de  $E$  est fini, le nombre de classe d'équivalence de  $R_L$  l'est aussi.  
( $E$  a au moins autant de classe d'équivalence que  $R_L$ ).

Soit donc  $E$  une relation d'équivalence vérifiant (2) et soit  $x, y \in \Sigma^*$  tel que  $x E y$ .

Pour  $\forall z \in \Sigma^*$ ,  $xz Eyz$  (car  $E$  est invariable à droite), c'est à dire que  $xz$  et  $yz$  appartiennent à la même classe.

Donc  $xz \in L$ ssi  $yz \in L$ , c'est-à-dire  $(x, y) \in R_L$ .

- $(3) \Rightarrow (1)$ : Nous allons définir un automate fini déterministe  $M_L$  tel que  $L(M_L) = L$ . Pour  $x \in \Sigma^*$  mettons  $[x] = [x]_{R_L}$ . Soit  $M = (Q, \Sigma, \delta, s, F)$  défini par:

$$\begin{aligned} Q &= \{[x] \mid x \in \Sigma^*\} & S &= [x] \\ F &= \{[x] \mid x \in L\} \\ \delta([x], a) &= [xa] \end{aligned}$$

On prouve maintenant que la définition est bonne et que  $L(M) = L$ .

1. Si  $[x] = [v]$  alors  $xz \in L$  ssi  $yz \in L$ ,  $\forall z \in \Sigma^*$   
Donc  $\forall a \in \Sigma$ ,  $xaz \in L$  ssi  $ya \in L$ ,  
et avec  $[xa] = [ya]$ .
2. On observe que  $\hat{\delta}([x], y) = [xy]$   $\forall x, y \in \Sigma^*$ , et donc  $\hat{\delta}(s, w) = \hat{\delta}([\epsilon], w) = [w]$ . On a alors que  $w \in L(M)$  ssi  $[w] \in F$  ssi  $w \in L$ .

Le théorème de Myhill-Nerode nous permet de construire des automates minimalistes pour un langage régulier.

- Utilisation du théorème de MN:

1. Prouvez qu'un langage est Régulier:  
On montre que  $R$  a un nombre fini de classe:
  - ① Donner les classes :  $C_1, \dots, C_k$
  - ② Prouver que chaque classe est une partition de  $\Sigma^*$ :

$$\bigcup_{i=1}^k C_i = \Sigma^* \text{ et } C_i \cap C_j = \emptyset \text{ si } i \neq j$$

2. Prouvez qu'un langage n'est Pas Régulier:  
On montre qu'il existe un nombre infini de classes d'équivalence.

# Grammaire Hors-Contexte

Définition:

Une grammaire hors-contexte (GHC) est un quadrupl (V, Σ, R, S), on définit :

- V : Un ensemble fini de variable
- Σ : Un alphabet ( $V \cap \Sigma = \emptyset$ )
- S : Un symbole de départ
- R : Un ensemble fini de règles (aussi appelé productions) de la forme  $A \rightarrow \alpha$ , avec  $\alpha \in (V \cup \Sigma)^*$

Les règles (R) permettent de remplacer une variable par le mot  $\alpha$  à droite de la règle.

Le langage généré par une grammaire sera l'ensemble des mots sur Σ auquel on parvient par une suite d'application de règles.

ex:  $L = \{ a^n b^n \mid n \in \mathbb{N} \}$

$$\begin{aligned} V &= \{ S \} & \Sigma &= \{ a, b \} \\ R &= S \rightarrow aSb \mid \epsilon \end{aligned}$$

Cette grammaire génère donc:

$$S \rightarrow aSb \rightarrow aasbb \rightarrow \dots \rightarrow a^n S b^n \rightarrow a^n b^n$$

Généralisation:

Soit  $G = (V, \Sigma, S, R)$ , soit  $u, v \in (V \cup \Sigma)^*$ .  
On définit:  $u$  donne  $v$  en  $k$  étapes  
(noté  $u \xrightarrow{k} v$ ) récursivement.

Cas de base:  $u \xrightarrow{0} u$

Pas de récurrence:  $u \xrightarrow{k} v$ ,  $\exists u_1, u_2, w \in (V \cup \Sigma)^*$

$$\text{tq: } \begin{aligned} u &\xrightarrow{k-1} u_1 A u_2 \\ A &\rightarrow w \quad \epsilon \in R \\ v &\rightarrow u_1 w u_2 \end{aligned}$$

On dit que  $u$  donne  $v$ :  $u \xrightarrow{*} v$ ,  $\exists k \in \mathbb{N}$  tq  $u \xrightarrow{k} v$ .

On définit le langage de la grammaire G comme  $L(G) = \{w \in \Sigma^* \mid s \xrightarrow{*} w\}$  qui est aussi hors-context.

## Langage Hors-Contexte :

- Définition : Un langage L est hors context si il existe une grammaire hors-context qui le génère, tel que  $L = L(G)$

- Fermeture des langages hors-context.

Soit  $L_1, L_2$  deux langage hors-context. Sachant que ces langages sont hors-context, alors il existe deux grammaire hors context qui les génère

- Union :  $L_1 \cup L_2 \in \mathcal{L}_{HC}$  ?

Supposons

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, R, \cup R_2 \cup \{S \xrightarrow{*} S_1 | S_2\})$$

$$\left. \begin{array}{l} S \xrightarrow{*} S_1 \xrightarrow{*} w \in L(G_1) \\ S \xrightarrow{*} S_2 \xrightarrow{*} w \in L(G_2) \end{array} \right\} S \notin V_1 \cup V_2$$

On aurait alors que  $L(G) = L(G_1) \cup L(G_2)$   
ssi  $V_1 \cap V_2 = \emptyset$  (Ne s'intersecte pas)

Donc  $L_1 \cup L_2 \in \mathcal{L}_{HC}$

- Concaténation :  $L_1 \cdot L_2 \in \mathcal{L}_{HC}$  ?

On crée  $G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, R)$  avec  $S \in V_1 \cup V_2$  et  $R = R_1 \cup R_2 \cup \{S \xrightarrow{*} S_1 \cdot S_2\}$  et on suppose que  $V_1 \cap V_2 = \emptyset$ .

Avec ceci, on aura que

$$S \xrightarrow{*} S_1 S_2 \xrightarrow{*} w_1 w_2, \text{ où } S_1 \xrightarrow{*} w_1, S_2 \xrightarrow{*} w_2 \text{ et } w_1 \in L_1, w_2 \in L_2. \quad \bullet L \in \mathcal{L}_{HC}$$

- Kleene Star:  $L^* \in \mathcal{L}_{HC}$  ?

On crée  $G = (V, \Sigma, S_1, R, \cup \{S_1 \xrightarrow{*} S, S_1 | S_1 \xrightarrow{*} \epsilon\})$

$$\begin{array}{c} S \xrightarrow{*} S, S_1 \xrightarrow{*} w_1 \cdot w_2 \\ \hookrightarrow \underbrace{S_1 S_1 \dots S_1}_K \xrightarrow{*} w_1 w_2 \dots w_K \end{array}$$

ayons déjà  $S_1 \xrightarrow{*} w$ , la preuve complète montrerait que

$$L^* \subseteq L(G)^*$$

- $L_1 \cap L_2$  et  $\overline{L}$ : Ne sont pas toujours hors context.

$$\text{ex: } L_1 = \{a^n b^n c^m \mid n, m \in \mathbb{N}\} \in \mathcal{L}_{HC}$$

$$L_2 = \{a^m b^n c^n \mid n, m \in \mathbb{N}\} \in \mathcal{L}_{HC}$$

$$L_1 \cap L_2 = \{a^n b^n c^n \mid n \in \mathbb{N}\} \notin \mathcal{L}_{HC}$$

voir pompiste H-C.

## Lemme Pompiste Hors-Context

Tout comme le pompiste pour les langages réguliers, ce théorème prouve qu'un langage n'est pas hors-context mais n'est pas suffisant pour prouver qu'il l'est.

Théorème: Si  $L \in \mathcal{L}_{HC}$ , alors  $\exists p \quad \forall w \in L, |w| \geq p$ ,  
 $\exists u, v, x, y, z \in \Sigma^*$  tq:

$$1 - w = uvxyz$$

$$2 - |vxy| \leq p$$

$$3 - |vy| > 0$$

$$4 - \forall i, uv^i xy^i z \in L$$

On ne choisit pas la décomposition. Pour un mot  $w$ , on doit tester toute les décomposition possible.

Pour prouver que  $L \notin \mathcal{L}_{HC}$  :

- On suppose que  $L \in \mathcal{L}_{HC}$
- On a  $\Delta \in \mathbb{N}$
- On choisit  $w \in L$ ,  $|w| > p$
- On prouve la contraposée du théorème:

$\forall p \in \mathbb{N}, \exists w \in L \text{ tq } \forall u, v, x, y, z \in \Sigma^*$  :

- $w = uvxxyz$
- $|vxy| \leq p$
- $|vy| > 0$
- $\exists i, ux^iy^iz \notin L$

## Forme Normal de Chomsky :

Algorithme pour transformé une  $G_{HC}$  en FNC :

1 - Etape optionnelle: On enlève les règles qui sont inutile.

Une règle ou une "sous-règle" est inutile si on ne peut pas l'atteindre via la variable de départ, ou si elle ne sera jamais terminal.

2 - Ajouté une variable initial  $S_0$  et Ajouté une règle  $S_0 \rightarrow S$ .

3 - Enlever les règles de la forme  $A \rightarrow \epsilon$  et modifier toutes les règles qui peuvent être impactez par  $A = \epsilon$  en ajoutant les possibilité générable par cette règle.

On modifie les règles atteinte en créant une nouvelle option dans l'éventualité où  $A \rightarrow \epsilon$

ex:  $A \rightarrow \epsilon \quad \left. \begin{array}{l} A \rightarrow a \\ B \rightarrow aa \end{array} \right\} \quad \text{devient} \quad \left. \begin{array}{l} A \rightarrow a \\ B \rightarrow aa \end{array} \right\} \quad B \rightarrow aa | aaa$

4 - Enlever les règles de la forme  $A \rightarrow B$  en ajoutant les règles de  $B$  au règles de  $A$ .

5 - Enlever les règles de la forme:  $A \rightarrow u_1 \dots u_k$ , pour  $k > 2$ .

On crée des nouvelle variables qui réfère au diverse variable possibles qui possède un max de 2 symbole.

6 - Enlever les règles de la forme:  $A \rightarrow Bx$ ,  $A \rightarrow xB$ ,  $A \rightarrow xy$ .

On crée des nouvelles règles qui réfère au symbole indépendamment.

# Machine de Turing:

Définition: Une machine de Turing (MT)

$M = (Q, \Sigma, \Gamma, \delta, S, q_A, q_R)$  est défini comme:

- $Q$ : Un ensemble fini d'état
- $\Sigma$ : Un alphabet d'entrée
- $\Gamma$ : Un alphabet du ruban ( $\Sigma \subseteq \Gamma$ ) et un symbole "Blanc"  $B \in \Gamma \setminus \Sigma$  et d'autres.
- $S \in Q$  un état initial.
- $q_A$ : un état acceptant
- $q_R$ : un état Refusant
- $\delta$ : une fonction de transitions.

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{\text{Droite, Gauche}\}$$

$$\text{Interprétation: } \delta(q, a) = (p, B, D)$$

"En lisant  $a$  sur le ruban, la machine se trouve dans l'état  $q$ , écrit  $B$  à la place de  $a$  (le remplace) sur le ruban. Elle passe ensuite dans un état  $p$  et se déplace d'une case dans une direction  $D$  ( $D \in \{\text{droite, gauche}\}$ ).

Une machine de Turing qui entre dans l'état  $q_A$  ou  $q_R$  s'arrête en acceptant ( $q_A$ ) ou refusant ( $q_R$ ) le mot d'entrée.

Décide →

On dit qu'une machine de Turing décide son langage si  $\forall w \in \Sigma^*$ , la machine s'arrête.

Informellement:  $L(MT)$  est l'ensemble des mots sur  $\Sigma^*$ .

La machine s'arrête car elle accepte ou refuse le mot.

reconnait →

Une MT reconnait un langage si elle s'arrête sur tous les mots du langage de  $L$  en l'acceptant, ou le refusant.

Note: Si  $L$  est un langage décidé par une MT:  $M$ , alors, il est également reconnue par  $M$ .

Informellement :  $\text{AL}_R$  est décider par une MT.

Un AF (Automate Fini) est une Machine de Turing où la tête de contrôle se déplace dans une unique direction.

Cependant, une MT ne possède qu'un seul état acceptant, et un refusant, tandis qu'un AF peut en avoir plusieurs.

**Calculer** → On dit qu'une MT à calculer la valeur d'une fonction  $F(a) = m$ , si elle s'arrête dans l'état  $q_A$ .

Une machine de Turing qui n'accepte ou refuse pas un mot peut se retrouver à boucler à l'infini.  
Nous pouvons cependant empêcher une MT de boucler à l'infini, mais dans certains cas, une boucle infini peut être nécessaire.