Mots :

- Alphabet Σ = Ensemble fini de symboles. Exemple : $\Sigma = \{a, b\}$.
- Σ^* : Ensemble de tous les mots de Σ. Eléments de Σ^* : $aba, abaabba$.
- Elément $\mathcal{E}$ = mot vide.
- **Mesure :** $w \in \Sigma^*$
  - Longueur du mot : $|w|$; Exemple : $|aba| = 3$ et $|\mathcal{E}| = 0$.
  - Nombre d'occurrences de a : $|w|_a$. Exemple : $|abaa|_a = 3$.
- **Opération :** $u, v, w \in \Sigma^*$
  - Concaténation : $v \cdot w$; Exemple : $ab \cdot bba = abbba$.
    - Non commutative : $a \cdot b \neq b \cdot a$;
    - Associative : $u \cdot (v \cdot w) = (u \cdot v) \cdot w$.
- **Relations :** $v, w \in \Sigma^*$
  - Préfixe de w : Il existe $u \in \Sigma^*$ tel que $w = v \cdot u$;
    - v Est préfixe de v ;
    - $\mathcal{E}$ Est préfixe de tout $v \in \Sigma^*$.
  - Suffixe de w : Il existe $u \in \Sigma^*$ tel que $w = u \cdot v$;
  - Facteur de w : Il existe $u_1, u_2 \in \Sigma^*$ tel que $w = u_1 \cdot v \cdot u_2$. Exemple : bb facteur de $abba$.
  - Sous-mots de w : v peut être obtenu de w par l'effacement de certaines lettres.
    Exemple abc sous-mot de $a\underline{ab}aab\underline{c}ab$.

Langages : L_1, L_2 2 langages

- Concaténation de 2 langages : $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$
  Exemple : $\{a, aa\} \cdot \{b, bb\} = \{ab, abb, aab, aabb\}$;
  - $\{\mathcal{E}\} \cdot L = L$; $\{a, \mathcal{E}\} \cdot \{b, bb\} = \{ab, abb, b, bb\}$;
  - $\emptyset \cdot L = L$. $\{a, aa\} \cdot \{a, aa\} = \{aa, aaa, aaaa\}$.
- Concaténation n de langage : si $n \in \mathbb{N}_0$, $L^n = L \cdot \ldots \cdot L$ (n fois).
- **Opérateurs ensemblistes :**
  - Union : $L_1 \cup L_2$;
  - Intersection : $L_1 \cap L_2$;
  - Complément par rapport à Σ^* : $L^{comp} = \Sigma^* - L$;
  - Etoile de Kleene : $L^* = L^0 \cdot L^1 \cdot L^2 \cdot \ldots$

Expression rationnelle ExpRat :

- **Syntaxe :**
  - Si $a \in \Sigma$, alors $a \in ExpRat$;
  - $\mathcal{E} \in ExpRat$; $\emptyset \in ExpRat$;
  - Si $r_1, r_2 \in ExpRat$, alors $(r_1 + r_2) \in ExpRat$;
  - Si $r_1, r_2 \in ExpRat$, alors $(r_1 \cdot r_2) \in ExpRat$;
  - Si $r \in ExpRat$, alors $r^* \in ExpRat$.
- **Sémantique :**
  - $\mathcal{L}(a) = \{a\}$; $\mathcal{L}(\mathcal{E}) = \{\mathcal{E}\}$; $\mathcal{L}(\emptyset) = \emptyset$;

- $\mathcal{L}(r_1 + r_2) = \mathcal{L}(r_1) + \mathcal{L}(r_2)$;
- $\mathcal{L}(r_1 \cdot r_2) = \mathcal{L}(r_1) \cdot \mathcal{L}(r_2)$;
- $\mathcal{L}(r^*) = \big(\mathcal{L}(r)\big)^*$.

- Equivalence : $\mathcal{L}(r_1) = \mathcal{L}(r_2)$ noté $r_1 \equiv r_2$;
- Langage L rationnel : $\exists\, r$ avec $L = \mathcal{L}(r)$;
- Lois d'équivalence pour toutes expressions r_1, r_2, r_3 :
  - $(r_1 \cdot r_2) \cdot r_3 \equiv r_1 \cdot (r_2 \cdot r_3)$;
  - $\mathcal{E} \cdot r_1 \equiv r_1$;
  - $r_1(r_2 + r_3) \equiv r_1 r_2 + r_1 r_3$.
  - $(r_1^*)^* \equiv r_1^*$;
  - $(a\ b)^* a \equiv a(b\ a)^*$.

Automate Déterministes :

- **Définition** : Un automate fini déterministe (AFD), noté $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$, consiste en :
  - Σ : Alphabet fini ;
  - Q : Ensemble fini d'états ;
  - q_0 : Etat initial ;
  - $F \subseteq Q$: Ensemble d'états acceptants ;
  - $\delta : Q \times \Sigma \nrightarrow Q$: Fonction de transition (lorsque l'on change d'état dans l'automate).
- AFD complet : Lorsque sa fonction δ est totale.
- Langage reconnu par l'automate $\mathcal{L}(\mathcal{A})$: $= \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$;
- Langage reconnaissable : Il existe un automate déterministe tel que $L = \mathcal{L}(\mathcal{A})$.

Automates non déterministes :

- **Définition :** Un automate non-déterministe (AFND) est $\mathcal{A} = (\Sigma, Q, Q_0, F, \delta)$, où :
  - Σ : Alphabet ;
  - Q : Ensemble fini d'états ;
  - $Q_0 \subseteq Q$: Ensemble des états initiaux ;
  - $F \subseteq Q$: Ensemble des états acceptants ;
  - $\delta : Q \times \Sigma \to P(Q) = \{\text{partie de } Q\}$.
- Déterminisation : Méthode pour transformer un AFND en AFD.

  Soit un AFND $\mathcal{A} = (\Sigma, Q, Q_0, F, \delta)$. On construit un ADF $\mathcal{A}' = (\Sigma, Q', q_0', F, \delta)$.
  - $Q' = P(Q)$;
  - $q_0' = Q_0$;
  - $F' = \{P \in Q' \mid P \cap F \neq \emptyset\}$;
  - $\delta'(P, a) = \bigcup_{p \in P} \delta(p, a)$ (l'ensemble des chemins de lettre a qui viennent de l'état q) ;
  - $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.

Automates non déterministes avec ε-transition :

- **Définition :** Un automate non-déterministe avec ε-transtition (AFNDE) est $(\Sigma, Q, I, F, \delta)$ où :

- Q un ensemble fini, d'états ;
- $I \subseteq Q$ les états initiaux ;
- $F \subseteq Q$ les états acceptants ;
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$.

- **Subtilité par rapport aux AFND :** Il est possible que des ε se trouve dans l'automate.
- **ε clôture** : $\overline{\{q\}} = \{q \in Q \mid \exists\, p \in P$ tel que on peut aller de q vers p seulement avec des ε transition $\}$.
- Méthode d'élimination des ε : Méthode pour transformer un AFNDE en AFND.

  Soit un AFNDE $\mathcal{A} = (\Sigma, Q, I, F, \delta)$. On construit un AFND $\mathcal{A}' = (\Sigma, Q', I', F', \delta)$.
  - $Q' = \{q \in Q \mid \exists\, a, p \text{ tq } \delta(q,a) = p\} \cup F$ (Tous les éléments qui ont un chemin sans ε) ;
  - $I' = \overline{\{I\}} \cap Q'$ (Tous les ε-clôture de l'état initial qui est dans Q') ;
  - $F' = F$;
  - $\delta'(q,a) = \overline{\delta(q,a)} \cap Q'$ (L'ε-clôture de la fonction δ de l'AFND qui sont dans Q').

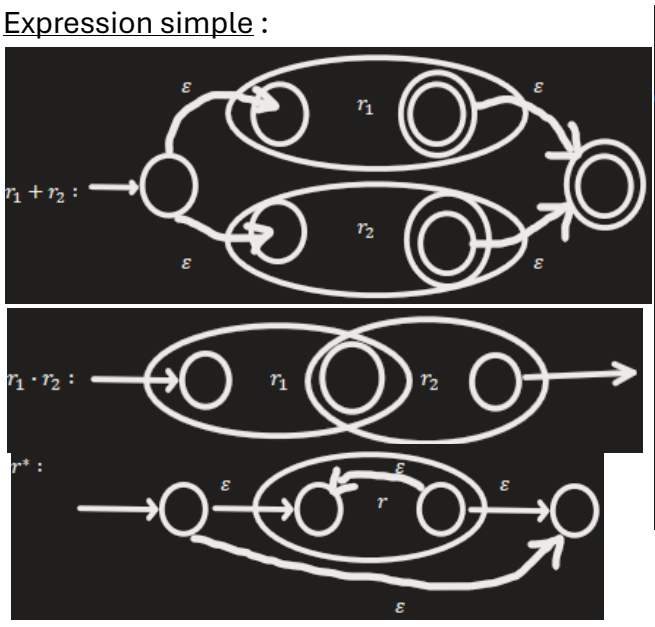
  On peut s'amuser ensuite à le déterminiser.

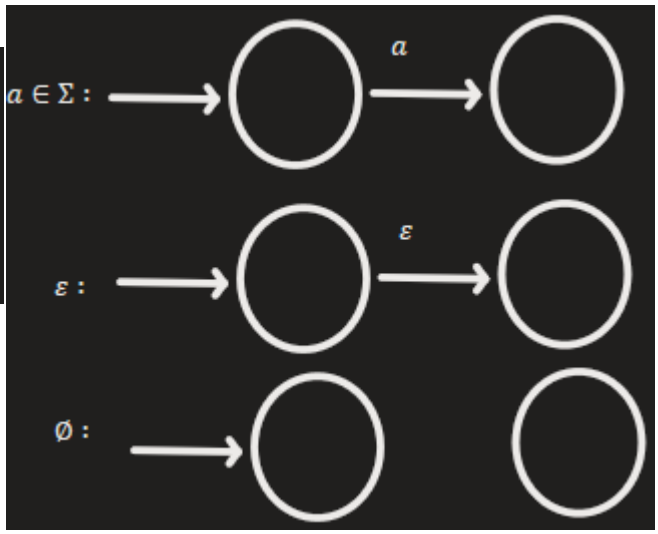
<u>Automate particuliers</u> :

- Automate qui reconnaît le **complémentaire** :
  - Ecrire un automate ;
  - Le mettre déterministe (si nécessaire) ;
  - Inversion les terminalité des états acceptants.
- **Miroir** $(\overline{\mathcal{L}})$**:** Si un automate $\mathcal{A}$ pour α :
  - On inverse la direction des flèches ;
  - On échange les initiaux avec les états terminaux ;

<u>Algorithme de Thompson</u> : Transformation d'une ExpRat en un AFNDE

- Propriété a respectée :
  - Un seul état initial ;
  - Un seul état acceptant, différents de l'état initial ;
  - Pas de transition vers l'état initial ;
  - Pas de transition sortant de l'état acceptant.
- <u>Expression simple</u> :





- **On applique ces propriétés expression simples sur les Expressions rationnelles**.
  On peut ensuite s'amuser à éliminer les ε puis peut être déterminiser.

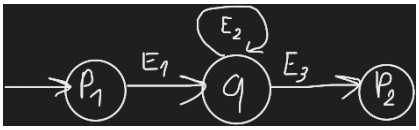
Algorithme de Glutchkov : Méthode pour transformer une ExpRat en AFND

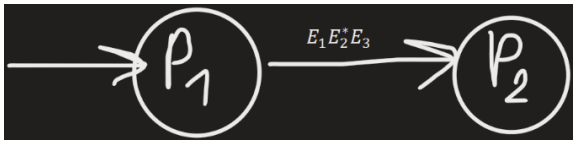
- Etape 1 = « Linéariser » l'expression : Mettre des numéros à chaque lettre pour les états ;
- Etape 2 = Analyse de l'expression :
  - Est-ce que on a ε dans l'expression ?
    Soit $eps : ExpRat \to \{true, false\}$ la fonction permettant de dire si ε est dans l'expression rationnelle
    - $eps(a) = false$; $eps(\varepsilon) = true$; $eps(\emptyset) = false$;
    - $eps(r_1 \cdot r_2) = eps(r_1) \wedge eps(r_2)$ (Et logique) ;
    - $eps(r_1 + r_2) = eps(r_1) \vee eps(r_2)$ (Ou logique) ;
    - $eps(r^*) = true$ Car r^* contient toujours ε.

    Si ε est dans l'expression, on la met comme état initial et acceptant ;
  - Trouver le $first$. $first(E) \stackrel{\text{def}}{=} \{a \in \Sigma' \mid qui\ peuvent\ appraître\ au\ début\ d'un\ mot\}$
    - $first(a) = \{a\}$; $first(\varepsilon) = \emptyset$; $first(\emptyset) = \emptyset$;
    - $first(r_1 + r_2) = first(r_1) \cup first(r_2)$;
    - $first(r_1 \cdot r_2) = \begin{cases} first(r_1) \text{ si } \neg eps(r_1) \\ first(r_1) \cup first(r_2) \text{ si } eps(r_1) \end{cases}$ (si il contient ε ou non) ;
    - $first(r^*) = first(r)$.
  - Trouver le $last$. $last(E) \stackrel{\text{def}}{=} \{a \in \Sigma' \mid qui\ peuvent\ appraître\ \text{à}\ la\ fin\ d'un\ mot\}$
    Elle a les mêmes propriétés que $first$.
  - Trouver le $next$. $next(E) \stackrel{\text{def}}{=} \{(x, y) \in \Sigma' \times \Sigma' \mid \exists\ y \text{ après un } x \text{ dans un mot de } \mathcal{L}(E)\}$
    Elle pour définition : $next(r_1 \cdot r_2) = next(r_1) \cup next(r_2) \cup last(r_1) \cdot first(r_2)$.
- Etape 3 : Construction
  - **Définition** : Automate de Glutchkov $(\Sigma, Q, I, F, \delta)$ avec :
    - $Q = \{0\} \cup \Sigma^*$;
    - $I = \{0\}$;
    - $F = last(r)$;
    - On a $\delta(0, a) = i$ quand $i \in first(r)$.
      Si $i \geq 0$, on a $\delta(i, a) = j$ quand $ij \in next(r)$.
  - Deux méthodes pour construire cette automate :
    - **Dessin de l'automate :**
      - Un état initial I ;
      - Un état par lettre ;
      - Si $x \in first(E)$, alors on ajoute $(I) \stackrel{X}{\to} (X)$;
      - Si $x \in last(E)$, alors (X) est terminal.
      - Pour chaque couple $(x, y) \in next$, alors $(X) \stackrel{Y}{\to} (Y)$;
    - **Tableau de l'automate :**
      Chaque ligne les différents états linéariser dans la 1 ;
      Chaque colonne représente une lettre de l'alphabet Σ.
      On y place les différents états par rapport à son prochain ($next$).

Algorithme de Brozowski et McClusky : Elimination des états

- Condition : Un seul état initial et un seul état acceptant ;
- Méthode :
  - On ajoute un état initial qui liera tous les anciens états initiaux ;
  - On ajoute un état terminal qui sera lié par tous les anciens états acceptants.
  - On va éliminer tous les états qui ne sont pas acceptants de la manière suivante :



→


Si E_2 ne contient rien, on aura seulement E_1E_3.

- Astuce : Toujours éliminer en priorité ceux avec le moins de flèches entrantes et sortantes.

Lemme d'Arden :

- **Substitution des variables par leurs définitions :**

  $subst(i,j) =$ Substitution de la variable L_i dans l'équation pour L_j.
- **Lemme d'Arden :** L'équation $L = A \cdot L + B$
  1) Sa solution minimale est A^*B ;
  2) C'est la seule solution quand $\varepsilon \notin \mathcal{L}(A)$.

Langage non rationnel :

- **Lemme de l'Etoile :**

  Si L est rationnel, alors $\exists\, N \in \mathbb{N}^*$ tel que $\forall\, w \in L$ avec $|w| \geq N$
  $\exists\, x, y, z$ tel que $w = x \cdot y \cdot z$ avec $|y| > 0$ et $|xy| \leq N$
  tel que $\forall\, k \in \mathbb{N}$ on a $x \cdot y^k \cdot z \in L$.
- Avec le lemme de l'Etoile : Il faut montrer la négation

  $\forall\, N \in \mathbb{N}^*, \exists\, w \in L$ avec $|w| \geq N$
  tel que $\forall\, x, y, z$ avec $w = x \cdot y \cdot z$ tel que $|y| > 0$ et $|xy| \leq N$
  $\exists\, k \in \mathbb{N}$ tel que $x \cdot y^k \cdot z \notin L$.
- Propriété de clôture :
  - Propriété de clôture sur les langages rationnels :
    - Union ∪ et Intersection ∪ ;
    - Concaténation · ;
    - Etoile de Kleene $*$;
    - Complémentaire ;
    - Différence ensembliste ;
    - Miroir ;
    - Préfixe.
  - A l'aide de ces propriétés, on peut montrer qu'un langage n'est pas rationnel par utilisation de ces propriété sur un langage rationnel ET non rationnelle.
    Soit L un langage. Pour montrer qu'il n'est pas rationnel, il faut montrer que L s'écrit de la forme $L = M\,.\,N$ avec . une opération de clôture tel que M est rationnel mais que N n'est pas rationnel.

Résiduels :

- Résiduel de L par rapport à w : $w^{-1}L = \{v \in \Sigma^* \mid wv \in L\}$;
- **Calcul pour les langages rationnel :**
  1) $a^{-1}\emptyset = \emptyset$;
  2) $a^{-1}\varepsilon = \emptyset$;
  3) $a^{-1}a = \varepsilon$; $a^{-1}b = \emptyset$ (où $b \in \Sigma, b \neq a$)
  4) $a^{-1}(r_1 + r_2) = a^{-1}r_1 + a^{-1}r_2$;
  5) $a^{-1}(r_1 \cdot r_2) = \begin{cases} (a^{-1}(r_1)) \cdot r_2 & \text{si } \neg eps(r_1) \\ (a^{-1}(r_1)) \cdot r_2 + a^{-1}r_2 \end{cases}$.
  6) $a^{-1}(r^*) = (a^{-1}r) \cdot r^*$.
- **L'automate des résiduels pour r :**
  - L'alphabet Σ ;
  - $Q = \{w^{-1} \cdot r \mid w \in \Sigma^*\}$;
  - $q_0 = r$;
  - $F = \{r' \in Q \mid eps(r')\}$;
  - $\delta(r', a) = a^{-1}r'$.

  Automate avec comme fonction de transition, la résiduels de la lettre.
- Simplification de certaines expressions :
  - $r + r \equiv r$;
  - $r_1 + r_2 \equiv r_2 + r_1$;
  - $(r_1 + r_2) + r_3 \equiv r_1 + (r_2 + r_3)$.

Relation d'équivalence induite par un langage :

- Relation d'équivalence $\sim$: Sur U
  - Réflexivité : $\forall\, x \in U, x \sim x$;
  - Symétrie : $\forall\, x, y \in U$, si $x \sim y$ alors $y \sim x$;
  - Transitivité : $\forall\, x, y, z \in U$, si $x \sim y$ et $y \sim z$ alors $x \sim z$.
- Relation d'équivalence induite d'un langage : Soit L un langage.

  $x \sim_L y$ si et seulement si $\forall\, w \in \Sigma^* : xw \in L \Leftrightarrow yw \in L$

  si et seulement si $x^{-1}L = y^{-1}L$.
- Classes d'équivalence $[x]_\sim$ sur U : $[x]_\sim = \{y \in U \mid x \sim y\}$.
- Indice $indice(\sim)$: Nombre de classes d'équivalence.

Théorème de Myhill-Nerode :

- $\sim_1$ Raffinement de $\sim_2$: Si et seulement si $x \sim_2 y \Rightarrow x \sim_1 y$.
- Propriété sur les indices : $indice(\sim_A) \leq |Q|$;
  - Quand l'automate contient des états non accessibles : $indice(\sim_A) < |Q|$.
  - Si $\sim_2$ est un raffinement de $\sim_1$, alors $indice(\sim_2) \geq indice(\sim_1)$.
  - Si $\mathcal{L}(A) = L$, alors $\sim_A$ est un raffinement de $\sim_L$.
  - Soit $x \sim_A y$, c'est-à-dire que $\delta^*(q_0, x) = \delta^*(q_0, y)$. Soit $w \in \Sigma^*$.

    $\delta^*(q_0, xw) = \delta^*(\delta^*(q_0, x), w) = \delta^*(\delta^*(q_0, y), w) = \delta^*(q_0, yw)$.

- **Théorème :** Soit $L \subseteq \Sigma^*$
  1) L est rationnel si et seulement si $indice(\sim_L)$ fini ;
  2) Si L est rationnel, alors $indice(\sim_L)$ est le nombre d'états du plus petit automate déterministes complet qui reconnaît L.

Minimisation d'automate :

1) Si A est un automate déterministes complet minimal qui reconnaît L,
   alors $\sim_A$ est identique à $\sim_L$.
2) L'automate minimal est complet. Si A_1, A_2 sont des automates déterministes complets d'un nombre minimal d'états qui reconnaissent L, les deux automates sont isomorphes (identiques aux noms des états prêt).
3) Critère pour qu'un automate déterministes complet A soit minimal.
   i) Tous les états de A sont accessibles.
   ii) Si $q_1, q_2 \in Q$ et $q_1 \neq q_2$: il existe w tel que $\delta^*(q_1, w) \in F$ et $\delta^*(q_2, w) \notin F$ (ou l'inverse).

   L'automate minimal à les propriétés : si i) et ii) : $\sim_L$ est un raffinement de $\sim_A$.
   Soit $x \sim_L y$. Supposons $x \sim_A y$, on a donc $\delta^*(q_0, x) \neq \delta^*(q_0, y)$.
   D'après ii) : il existe w tel que $\underbrace{\delta^*(\delta^*(q_0, x), w)}_{xw} \in F$ et $\underbrace{\delta^*(\delta^*(q_0, y), w)}_{yw} \notin F$

- **Méthode de Moore :**
  - On travaille avec des partitions de Q. $(P_1, \dots, P_n)$ est une partition de Q :
    - Tous les $P_i \subseteq Q$; Tous les $P_i \neq \emptyset$;
    - Tous les $P_i \cap P_j \neq \emptyset$ si $i \neq j$;
    - $Q = P_1 \cup \dots \cup P_n$.
  - On élimine avant, tous les états non accessibles ;
  - On commence par la partition $(F, Q - F)$.
  - On dit que $a \in \Sigma$ sépare P_i en $R_1, \dots, R_n$ si $R_j = \{q \in P_i \mid \delta(q, a) \in P_j\}$.
    Quand c'est le cas, on remplace P_i pour tous les R_j qui sont non-vides.
  - Construction de l'automate :
    - Etat minimal : $P_1, \dots, P_n$;
    - Etat initial : Le P_i avec q_0 ;
    - P_i est acceptant si $P_i \subseteq F$;
    - $\delta(P_i, a) = P_j$ si pour $q \in P_i$, $\delta(q, a) \in P_j$.