- Définitions :

	Signification	Exemple
Alphabet	Ensemble fini de lettres	$\Sigma = \{a, b\}$
Mot	Suite finie de lettre	$m = abaa$
ε	Mot vide (sans lettre)	
Langage	Ensemble de mots	$L = \{\varepsilon, a, baa\}$

ε est un mot, pas une lettre.

- Opérations sur des mots $u = u_1...u_n$ et $v = v_1...v_p$:

	Définition	Exemple avec $u = ab$ et $v = cbc$
Concaténation	$uv = u_1...u_nv_1...v_p$	$uv = abcbc$
Puissance	$u^n = u...u$	$u^3 = ababab$
Taille	$\|u\| = n$	$\|u\| = 2$

Deux mots sont égaux s'ils ont la même taille et les mêmes lettres.

- Opérations sur des langages $L_1 = \{\varepsilon, ab\}$ et $L_2 = \{b, ab\}$:

	Définition	Exemple
Concaténation	$L_1L_2 = \{uv \mid u \in L_1, v \in L_2\}$	$L_1L_2 = \{b, ab, abb, abab\}$
Puissance	$L^n = \{u^n \mid u \in L\}$	$L_1^2 = \{\varepsilon, ab, abab\}$
Etoile	$L^* = \bigcup_{k\in\mathbb{N}} L^k$	$L_1^* = \{\varepsilon, ab, abab...\}$

Comme L_1 et L_2 sont des ensembles, on peut aussi considérer $L_1 \cup L_2$, $L_1 \cap L_2$...

- Les langages réguliers sont tous ceux qu'on peut obtenir avec les règles suivantes :
  - Un langage fini est régulier
  - L_1 et L_2 réguliers $\implies$ $L_1 \cup L_2$ régulier
  - L_1 et L_2 réguliers $\implies$ L_1L_2 régulier
  - L régulier $\implies$ L^* régulier

  Exemples : Un alphabet Σ est toujours régulier car fini. Σ^* est régulier car est l'étoile du langage régulier Σ.

- Une expression régulière est une suite de symboles contenant : lettres, $\emptyset$, ε, $|$ (union, parfois notée $+$), *.
  À chaque expression régulière e on associe un langage $L(e)$.

  Exemple : Le langage de l'expression régulière $e = a^*b \mid \varepsilon$ est $L(e) = (\{a\}^*\{b\}) \cup \{\varepsilon\}$.

- L régulier $\iff$ $\exists$ une expression régulière de langage L.

Exemples de langages réguliers sur $\Sigma = \{a, b\}$:

1. Mots contenant au plus un a : $b^*(a|\varepsilon)b^*$.
2. Mots de taille $n \equiv 1 \mod 3$: $((a|b)^3)^*(a|b)$.
3. Mots contenant un nombre pair de a : $(ab^*a|b)^*$.
4. Mots contenant un nombre impair de a : $b^*a(ab^*a|b)^*$.

- Définition possible d'expression régulière en OCaml :

```
type 'a regexp =
    | Vide | Epsilon | L of 'a
    | Union of 'a regexp * 'a regexp
    | Concat of 'a regexp * 'a regexp
    | Etoile of 'a regexp

(* définition de e ci-dessus *)
let e = Union(Concat(Etoile(a), b), Epsilon)
```

Exemple : déterminer si ε appartient au langage de e.

```
let rec has_eps e = match e with
    | Vide | L _ -> false
    | Epsilon | Etoile _ -> true
    | Union(e1, e2) -> has_eps e1 || has_eps e2
    | Concat(e1, e2) -> has_eps e1 && has_eps e2
```

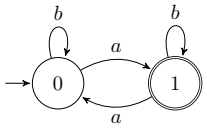
- Quelques techniques de preuve :
  - Sur des mots : récurrence sur la taille du mot.
  - Pour montrer l'égalité de deux langages : double inclusion ou suite d'équivalences.
  - Pour montrer $P(L)$ pour un langage régulier L : par induction ($\approx$ récurrence), en montrant le cas de base (si L est un langage fini) et les cas d'hérédité ($P(L_1) \wedge P(L_2) \implies P(L_1L_2)$, $P(L_1) \wedge P(L_2) \implies P(L_1 \cup L_2)$, $P(L) \implies P(L^*)$).
  - Pour montrer $P(e)$ pour une expression régulière e : par induction, en montrant les cas de base ($P(\emptyset)$, $P(\varepsilon)$, $P(a)$, $\forall a \in \Sigma$) et les cas d'hérédité ($P(e_1)$ et $P(e_2) \implies P(e_1e_2)$ et $P(e_1|e_2)$, $P(e) \implies P(e^*)$).

  Exemple : Le miroir d'un mot $u = u_1...u_n$ est $\widetilde{u} = u_n...u_1$ et le miroir d'un langage L est $\widetilde{L} = \{\widetilde{u} \mid u \in L\}$.
  Montrons : L régulier $\implies$ $\widetilde{L}$ régulier.
  On pourrait le montrer par récurrence, mais il est peut-être plus simple de définir une fonction $f(e)$ qui à une expression régulière e associe une expression régulière pour le miroir de $L(e)$:
  - $f(\emptyset) = \emptyset$, $f(\varepsilon) = \varepsilon$ et $\forall a \in \Sigma, f(a) = a$.
  - $f(e_1e_2) = f(e_2)f(e_1)$ (le miroir de uv est $\widetilde{v}\widetilde{u}$).
  - $f(e_1|e_2) = f(e_1)|f(e_2)$.
  - $f(e_1^*) = f(e_1)^*$.

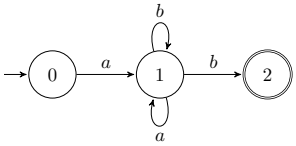
  On a bien défini une fonction f telle que, pour toute expression régulière e, $f(e)$ est une expression régulière de $\widetilde{L(e)}$. Donc le miroir d'un langage régulier est régulier.

- Un **automate** est un 5-uplet $A = (\Sigma, Q, I, F, E)$ où :
  - Σ est un alphabet.
  - Q est un ensemble fini d'**états**.
  - $I \in Q$ est un ensemble d'**états initiaux**.
  - $F \subseteq Q$ est un ensemble d'**états acceptants** (ou **finaux**)
  - $E \subseteq Q \times \Sigma \times Q$ est un ensemble de **transitions**.
    On peut remplacer l'ensemble E de transitions par une **fonction de transition** $\delta : Q \times \Sigma \longrightarrow \mathcal{P}(Q)$
- – Un chemin dans A est **acceptant** s'il part d'un état initial pour aller dans un état final.
  - Un mot est **accepté** par A s'il est l'étiquette d'un chemin acceptant.
  - Le **langage** $L(A)$ **accepté** (ou **reconnu**) par A est l'ensemble des mots acceptés par A.

  Exemple :
  - Soit A l'automate suivant :

    Alors $L(A) = \{$ mot avec un nombre impair de a $\}$ $= L(b^*a(b^*ab^*ab^*)^*)$.
  - Le langage $a(a+b)^*b$ est reconnaissable, car reconnu par l'automate ci-dessous.

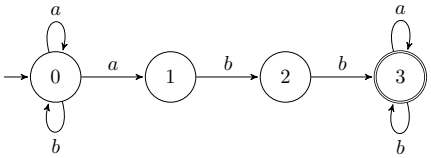
- Pour déterminer algorithmiquement si un automate A accepte un mot $u = u_1...u_n$, on peut calculer de proche en proche $Q_0 = I$, $Q_1 =$ états accessibles depuis Q_0 avec la lettre u_1, $Q_2 =$ états accessibles depuis Q_0 avec la lettre u_2... et regarder si Q_n contient un état final.
- Soit $A = (\Sigma, Q, I, F, E)$ un automate.
  - A est **complet** si : $\forall q \in Q,\ \forall a \in \Sigma,\ \exists (q, a, q') \in E$
  - Un automate $(\Sigma, Q, \{q_i\}, F, E)$ est **déterministe** si :
    1. Il n'y a qu'un seul état initial q_i.
    2. $(q, a, q_1) \in E \wedge (q, a, q_2) \in E \implies q_1 = q_2$: il y a au plus une transition possible en lisant une lettre depuis un état
  - Un automate déterministe et complet possède une unique transition possible depuis un état en lisant une lettre.
    On a alors une fonction de transition de la forme $\delta : Q \times \Sigma \longrightarrow Q$ qu'on peut étendre en $\delta^* : Q \times \Sigma^* \longrightarrow Q$ définie par :
    * $\delta^*(q, \varepsilon) = q$
    * Si $u = av$, $\delta^*(q, av) = \delta^*(\delta(q, a), v)$

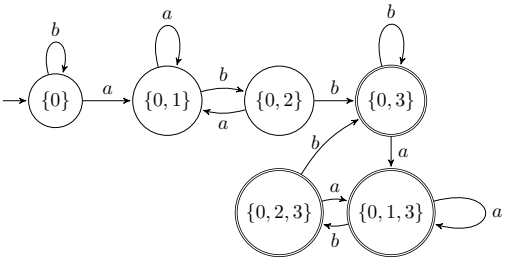
    Ainsi, $\delta^*(q, u)$ est l'état atteint en lisant le mot u depuis l'état q.
    On a alors $u \in L(A) \iff \delta^*(q_i, u) \in F$.
- Deux automates sont **équivalents** s'ils ont le même langage.
- Soit A un automate. Alors A est équivalent à un automate déterministe complet.

  Preuve : Utilise l'automate des parties $A' = (\Sigma, \mathcal{P}(Q), \{I\}, F', \delta')$ où $F' = \{X \subseteq Q \mid X \cap F \neq \emptyset\}$

  Remarque : Si on veut juste un automate complet (pas forcément déterministe), on peut ajouter un état poubelle vers lequel on redirige toutes les transitions manquantes. Dans l'automate des parties, cet état poubelle est $\emptyset$.

  Exemple : Un automate A avec son déterminisé A'.

- Soit L un langage reconnaissable. Alors $\overline{L}$ $(= \Sigma^* \backslash L)$ est reconnaissable.

  Preuve : Soit $A = (\Sigma, Q, q_i, F, \delta)$ un automate déterministe complet reconnaissant L. Alors $A' = (\Sigma, Q, q_i, Q \backslash F, \delta)$ (on inverse états finaux et non-finaux) reconnaît $\overline{L}$.
- Soient L_1 et L_2 des langages reconnaissables. Alors :
  - $L_1 \cap L_2$ est reconnaissable.
  - $L_1 \cup L_2$ est reconnaissable.
  - $L_1 \backslash L_2$ est reconnaissable.

  Preuve : Soient $A_1 = (Q_1, q_1, F_1, \delta_1)$ et $A_2 = (Q_2, q_2, F_2, \delta_2)$ des automates finis déterministes complets reconnaissants L_1 et L_2. Soit $A = (Q_1 \times Q_2, (q_1, q_2), F, \delta)$ (**automate produit**) où :
  - $F = F_1 \times F_2$: A reconnait $L_1 \cap L_2$.
  - $F = \{(q_1, q_2) \mid q_1 \in F_1 \text{ ou } q_2 \in F_2\}$: A reconnait $L_1 \cup L_2$.
  - $F = \{(q_1, q_2) \mid q_1 \in F_1 \text{ et } q_2 \notin F_2\}$: A reconnait $L_1 \setminus L_2$.

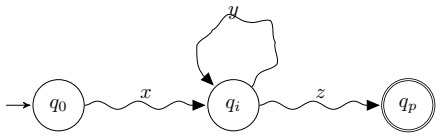
  Remarque : Comme l'ensemble des langages reconnaissables est égal à l'ensemble des langages rationnels, l'ensemble des langages rationnels est aussi stable par complémentaire, intersection et différence.
  Il n'y a pas de stabilité par inclusion (L rationnel et $L' \subseteq L$ n'implique pas forcément L' rationnel).

- (**Lemme de l'étoile** ♡) Soit L un langage reconnaissable par un automate à n états.
  Si $u \in L$ et $|u| \geq n$ alors il existe des mots x, y, z tels que :
  - $u = xyz$
  - $|xy| \leq n$
  - $y \neq \varepsilon$
  - $xy^*z \subseteq L$ (c'est-à-dire : $\forall k \in \mathbb{N},\ xy^kz \in L$)

  Preuve : Soit $A = (\Sigma, Q, I, F, \delta)$ un automate reconnaissant L et $n = |Q|$.
  Soit $u \in L$ tel que $|u| \geq n$.
  u est donc l'étiquette d'un chemin acceptant C :

  $$q_0 \in I \xrightarrow{u_0} q_1 \xrightarrow{u_1} ... \xrightarrow{u_{p-1}} q_p \in F$$

  C a $p + 1 > n$ sommets donc passe deux fois par un même état $q_i = q_j$ avec $i < n$. La partie de C entre q_i et q_j forme donc un cycle.

  Soit $x = u_0u_1...u_{i-1}$, $y = u_i...u_j$ et $z = u_{j+1}...u_{p-1}$.
  xy^kz est l'étiquette du chemin acceptant obtenu à partir de C en passant k fois dans le cycle.

  Application : $L_1 = \{a^nb^n \mid n \in \mathbb{N}\}$ n'est pas reconnaissable.
  Preuve : Supposons L_1 reconnaissable par un automate à n états. Soit $u = a^nb^n$. Clairement, $u \in L_1$ et $|u| \geq n$.
  D'après le lemme de l'étoile : il existe x, y, z tels que $u = xyz$, $|xy| \leq n$, $y \neq \varepsilon$ et $xy^*z \subseteq L$.
  Comme $|xy| \leq n$, x et y ne contiennent que des a : $x = a^i$ et $y = a^j$. Comme $y \neq \varepsilon$, $j > 0$.
  En prenant $k = 0$: $xy^0z = xz = a^{n-j}b^n$.
  Or $j > 0$ donc $a^{n-j}b^n \notin L_1$: absurde.

Automate de Glushkov : régulier $\implies$ reconnaissable

- Une expression régulière est **linéaire** si chaque lettre y apparaît au plus une fois : $a(d+c)^*b$ est linéaire mais pas $ac(a+b)$.

- Soit L un langage. On définit :
  - $P(L) = \{a \in \Sigma \mid a\Sigma^* \cap L \neq \emptyset\}$ (premières lettres des mots de L)
  - $S(L) = \{a \in \Sigma \mid \Sigma^* a \cap L \neq \emptyset\}$ (dernières lettres des mots de L)
  - $F(L) = \{u \in \Sigma^2 \mid \Sigma^* u \Sigma^* \cap L \neq \emptyset\}$ (facteurs de longueur 2 des mots de L)
  - L est **local** si, pour tout mot $u = u_1u_2...u_n \neq \varepsilon$:

    $$u \in L \iff u_1 \in P(L) \wedge u_n \in S(L) \wedge \forall k, u_k u_{k+1} \in F(L)$$

    Il suffit donc de regarder la première lettre lettre, la dernière lettre et les facteurs de taille 2 pour savoir si un mot appartient à un langage local.
    Remarques :
    * $\implies$ est toujours vrai donc il suffit de prouver $\impliedby$.
    * Définition équivalente :

      $$L \text{ local} \iff L \setminus \{\varepsilon\} = (P(L) \cap S(L)) \setminus N(L)$$

      où $N(L) = \Sigma^2 \setminus F(L)$.

    Exemples :
    - Si $L_2 = (ab)^*$ alors $P(L_2) = \{a\}$, $S(L_2) = \{b\}$ et $F(L_2) = \{ab, ba\}$. De plus si $u = u_1u_2...u_n \neq \varepsilon$ avec $u_1 \in P(L), u_n \in S(L)$, et $\forall k, u_k u_{k+1} \in F(L)$ alors $u_1 = a$, $u_n = b$ et on montre (par récurrence) que $u = abab...ab \in {}_2$. Donc L_2 est local.
    - Si $L_3 = a^* + (ab)^*$ alors $P(L_3) = \{a\}$, $S(L_3) = \{a, b\}$, $F(L_3) = \{aa, ab, ba\}$. Soit $u = aab$. La première lettre de u est dans $P(L_3)$, la dernière dans $S(L_3)$ et les facteurs de u sont aa et ba qui appartiennent à $F(L_3)$. Mais $u \notin L_3$, ce qui montre que L_3 n'est pas local.

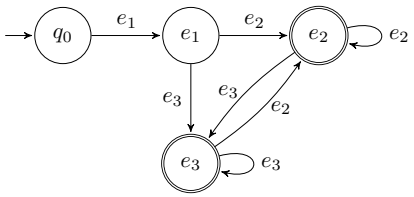
- Un automate déterministe (Σ, Q, q_0, F, E) est **local** si toutes les transitions étiquetées par une même lettre aboutissent au même état : $(q_1, a, q_2) \in E \wedge (q_3, a, q_4) \in E \implies q_2 = q_4$

- Un langage local L est reconnu par un automate local.

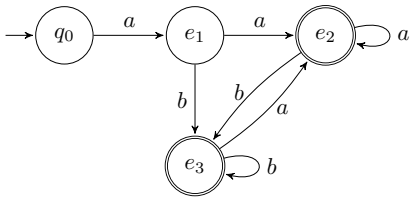
  Preuve : L est reconnu par (Σ, Q, q_0, F, E) où :
  - $Q = \Sigma \cup \{q_0\}$: un état correspond à la dernière lettre lue
  - $F = S(L)$ si $\varepsilon \notin L$, sinon $F = S(L) \cup \{q_0\}$.
  - $E = \{(q_0, a, a) \mid a \in P(L)\} \cup \{(a, b, b) \mid ab \in F(L)\}$

- L'**algorithme de Berry-Sethi** permet de construire un automate à partir d'une expression régulière e.

  Exemple avec $e = a(a+b)^*$:
  1. On linéarise e en e', en remplaçant chaque occurrence de lettre dans e par une nouvelle lettre : $e' = e_1(e_2 + e_3)^*$
  2. On peut montrer que $L(e')$ est un langage local.
  3. Un langage local est reconnu par l'automate local $A = (\Sigma, Q, q_0, F, E)$

  4. On fait le remplacement inverse de 1. sur les transitions de A pour obtenir un automate reconnaissant $L(e)$:

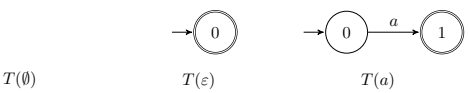
Automate de Thompson : régulier $\implies$ reconnaissable

- Une ε-transition est une transition étiquetée par ε.

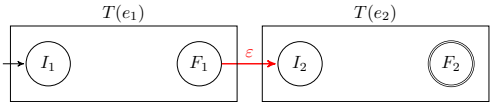
- Un automate avec ε-transitions est équivalent à un automate sans ε-transitions.

  Preuve : Si $A = (\Sigma, Q, I, F, \delta)$ est un automate avec ε-transitions, on définit $A' = (\Sigma, Q, I', F, \delta')$ où :
  - I' est l'ensemble des états atteignables depuis un état de I en utilisant uniquement des ε-transitions.
  - $\delta'(q, a)$ est l'ensemble des états q' tel qu'il existe un chemin de q à q' dans A étiqueté par un a et un nombre quelconque de ε (ce qui peut être trouvé par un parcours de graphe).

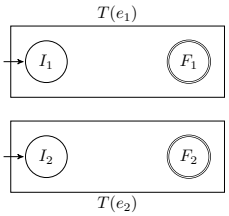
- L'**automate de Thompson** est construit récursivement à partir d'une expression régulière e :
  - Cas de base :

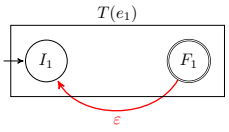
  - $T(e_1e_2)$: ajout d'une ε-transition depuis chaque état final de $T(e_1)$ vers chaque état initial de $T(e_2)$.

  - $T(e_1|e_2)$: union des états initiaux et des états finaux.



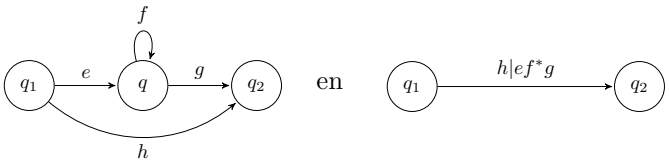
– $T(e_1^*)$: ajout d'une ε-transition depuis chaque état final vers chaque état initial.



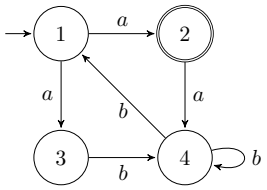
Élimination des états : reconnaissable $\Longrightarrow$ régulier

- Tout automate est équivalent à un automate avec un unique état initial sans transition entrante et un unique état final sans transition sortante.
  Preuve : On ajoute un état initial q_i et un état final q_f et des transitions ε depuis q_i vers les états initiaux et depuis les états finaux vers q_f.

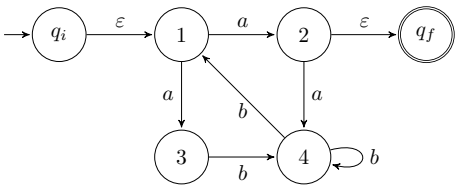
- **Méthode d'élimination des états** : On considère un automate A comme dans le point précédent. Tant que A possède au moins 3 états, on choisit un état $q \notin \{q_i, q_f\}$ et on supprime q en modifiant les transitions :



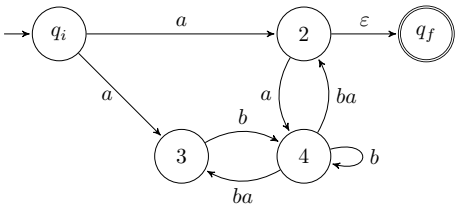
Exemple :



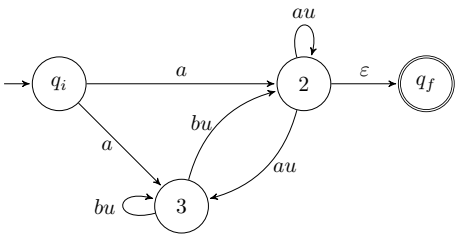
1. On commence par se ramener à un automate avec un état initial sans transition entrante et un état final sans transition sortante :



2. Suppression de l'état 1 :

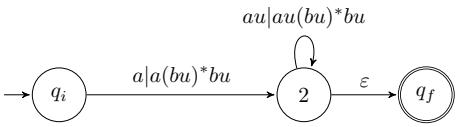


3. Suppression de l'état 4 :

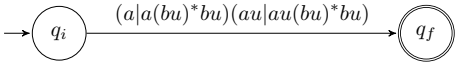


Avec $u = b^*ba$.

4. Suppression de l'état 3 :



5. Suppression de l'état 2 :



On obtient l'expression régulière $a|a(bu)^*bu(au|au(bu)^*bu)$, où $u = b^*ba$.

régulier $\Longleftrightarrow$ reconnaissable

- **Théorème de Kleene** : un langage est régulier si et seulement si il est reconnaissable par un automate.

- Les théorèmes sur les automates s'appliquent aussi aux langages réguliers, et inversement. Notamment, les langages réguliers sont stables par union, concaténation, étoile, intersection, complémentaire, différence.