Automates

Automates à états finis

Damien Nouvel

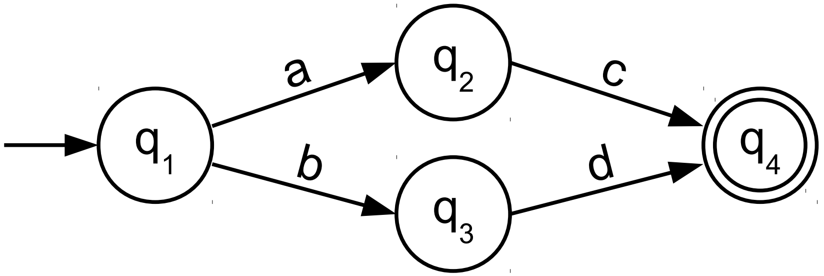
Plan

- Représentation des automates (FSA)
- Définition formelle (DFA)
- Déterminisme du FSA (DFA / NFA / ε-NFA)
- Équivalence DFA / NFA / ε-NFA

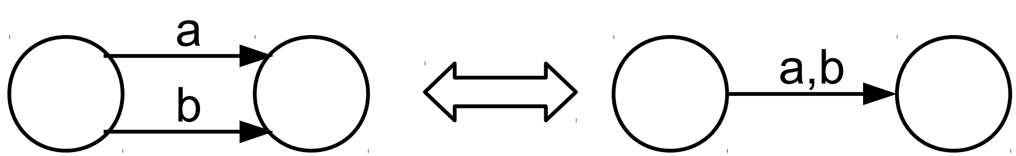
- Représentation des automates (FSA)
- Définition formelle (DFA)
- Déterminisme du FSA (DFA / NFA / ε-NFA)
- Équivalence DFA / NFA / ε-NFA

- Automate à états finis
  - **Finite State Automaton (FSA)**
  - Machine abstraite, issue des travaux de **A. Turing**
- Éléments de l'automate
  - Une ensemble fini d'**états** possibles
  - Un ensemble fini de **symboles** en entrées
  - Une fonction de **transition** entre états
- Représentations
  - **Diagrammes de transition**
  - **Tables de transition**
  - **Notations formelles**

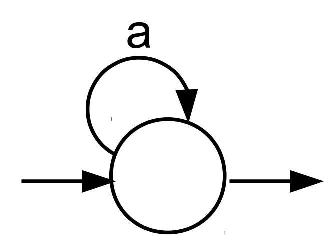
- Diagrammes de transition
  - Représentation « graphique »
  - Graphe orienté étiqueté :
    - **Nœuds** : **états** de l'automate
      - Étiqueté par les noms des **états** (généralement q_x : q_1, q_2, q_3, ...)
      - État **final** : **double cercle**
    - **Arcs orientés** : **transitions** de l'automate
      - Étiqueté par les **symboles** (a, b, c, d, ...)
      - État **initial** : marqué par un **arc sans nœud d'origine**



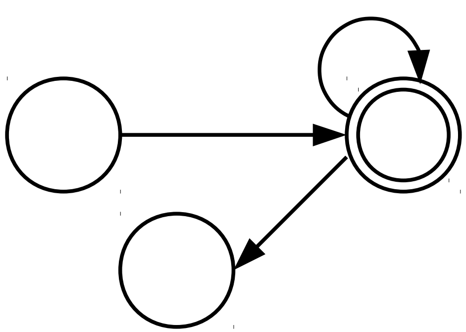
- Diagrammes de transition (suite)
  - Pour la lisibilité, lorsque deux arcs de même orientation sont possibles entre deux nœuds, ils sont fusionnés (**disjonction**) :



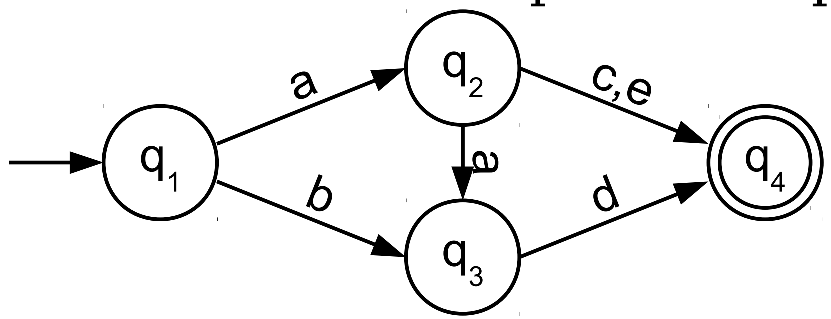
  - Un arc peut « boucler » un état **sur lui-même** :



  - Des transitions peuvent **partir de l'état final** :

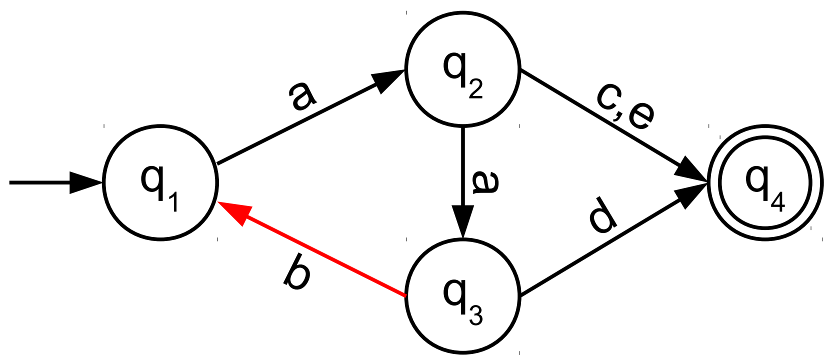


- Diagrammes de transition (suite)
  - Les **transitions** correspondent à la **concaténation**
  - On « suit » les arcs pour voir quel langage est accepté :



L = {ac, ae, bd, aad }

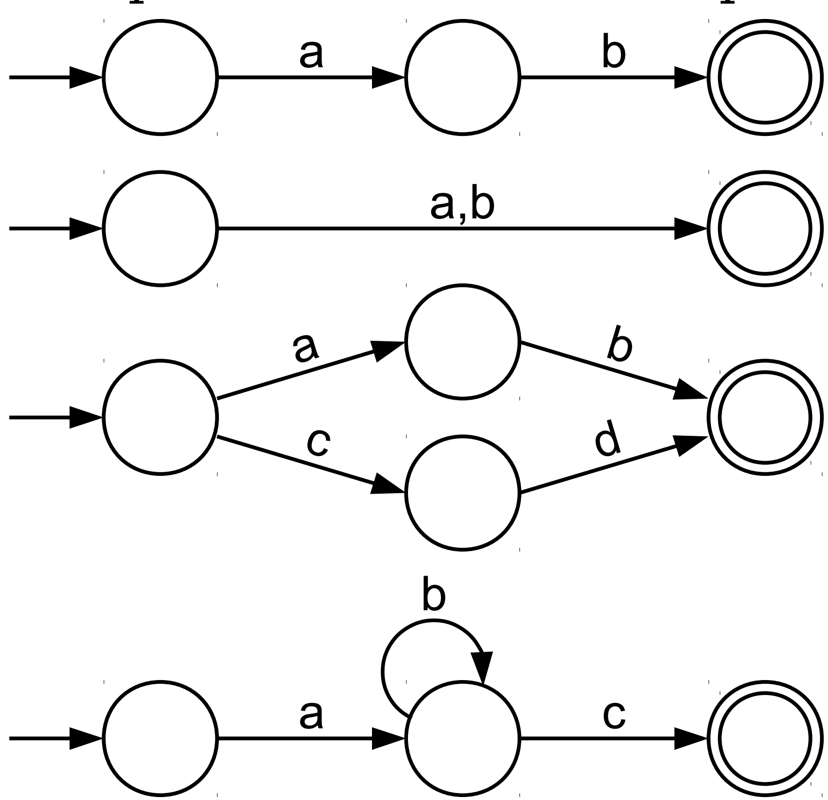
  - Le langage n'est pas forcément fini :



L = {ac, ae, aad, aabac, aabae, aabaad, aabaabac, aabaabae, aabaabaad ... }

Automates à états finis
Représentation des automates (FSA)

- Diagrammes de transition (suite)
  - Peuvent être mis intuitivement (formellement aussi) en correspondance avec des expressions régulières



R = ab

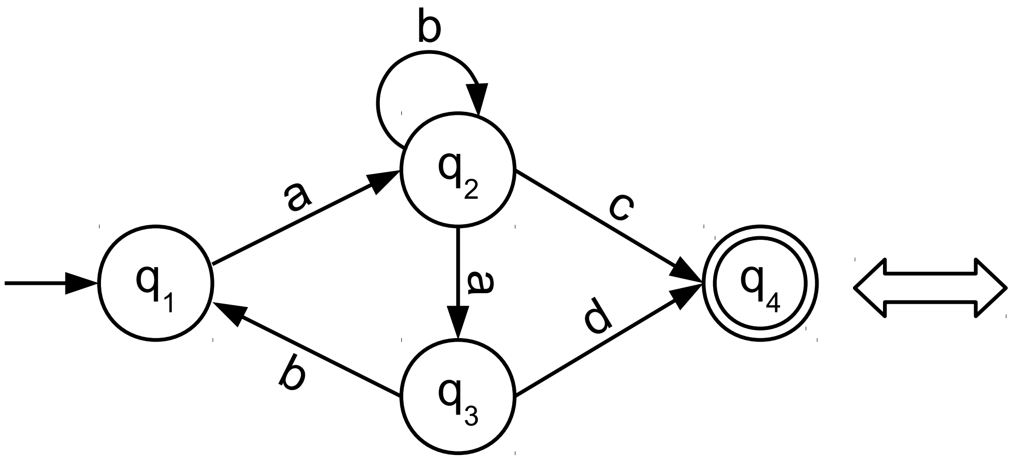
R = a+b

R = ab+cd

R = $ab^{*}c$

Automates à états finis
Représentation des automates (FSA)

- Table de transitions
  - **Équivalente** à la représentation par diagramme de transitions
  - **États** en **lignes**, **symboles** en **colonnes**
    - Ligne de l'état **initial** marqué par une **flèche →**
    - Ligne de l'état **final** marqué par une **étoile** *
  - **Contenu** décrit les **transitions** depuis un état / symbole



	a	b	c	d
→ **q_1**	q_2	∅	∅	∅
q_2	q_3	q_2	q_4	∅
q_3	∅	q_1	∅	q_4
* **q_4**	∅	∅	∅	∅

- Représentation des automates (FSA)
- Définition formelle (DFA)
- Déterminisme du FSA (DFA / NFA / ε-NFA)
- Équivalence DFA / NFA / ε-NFA

Automates à états finis
Définition formelle (DFA)

- Notations pour un automate « **déterministe** » **(DFA)**
  - Un ensemble **fini** d'**états**
    - Notation : $Q = \{q_1, q_2, q_3 \ldots\}$
  - Un ensemble **fini** de **symboles** (**alphabet**)
    - Notation : $\Sigma = \{a, b, c, d \ldots\}$
  - Une **fonction de transition** qui prend en paramètre un état et un symbole et qui renvoie un état
    - Notation : $\delta(q_i, a) = q_j$ (« signature » $\delta : Q \times \Sigma \rightarrow Q$)
  - Un **état initial**
    - Notation : $q_0 \in Q$
  - Un ensemble d'**états finaux**
    - Notation : $F \subseteq Q$

Automates à états finis
Définition formelle (DFA)

- **Automate à états finis**
  - Défini comme le quintuplet : $A = (Q, \Sigma, \delta, q_0, F)$
  - Définition **équivalente** aux diagrammes et tables de transition
  - Repose sur la définition de la **fonction de transition δ**, souvent définie par extension (en listant les cas possibles)
    - La fonction peut renvoyer Ø (par ex. $\delta(q_1, d) = \emptyset$)
  - Définition de la **fonction de transition « étendue »**
    - Fonction qui prend un mot en entrée et utilise δ de l'automate
    - Notation ($\alpha \in \Sigma^*$) : $\overline{\delta}(q_i, \alpha) = q_j$ (« signature » $\overline{\delta} : Q \times \Sigma^* \rightarrow Q$)
    - Définie **récursivement** par **décomposition** du mot w :
      - Si $w = \varepsilon$ alors $\overline{\delta}(q_i, \varepsilon) = \{ q_i \}$
      - Si $w = x.a$ tels que et $x \in \Sigma^*$ et $a \in \Sigma$ alors $\overline{\delta}(q_i, w) = \delta(\overline{\delta}(q_i, x) , a)$

- **Langage reconnu** par un automate déterministe
  - Ensemble des mots tels que la fonction de transition étendue appliquée à l 'état initial et au mot conduit à un état final :
    - $L(A) = \{ w \in \Sigma^* \mid \overline{\delta}(q_0, w) \in F \}$
  - **Théorème de Kleene** : par définition, le langage reconnu par un automate à états finis (NFA) est **régulier** (ou rationnel)
- Avantages et limites des automates à états finis (**FSA**)
  - **Algorithmes rapides** pour implémenter les FSA
  - Impossible de « compter » dans le cas général
    - Pas d'automate qui décrive un mot ayant autant de a que de b
    - Pas d'automate pour reconnaître une expression arithmétique bien formée (nécessité de recourir à des grammaires)

- Représentation des automates (FSA)
- Définition formelle (DFA)
- Déterminisme du FSA (DFA / NFA / ε-NFA)
- Équivalence DFA / NFA / ε-NFA

Automates à états finis
Déterminisme du FSA (DFA / NFA / ε-NFA)

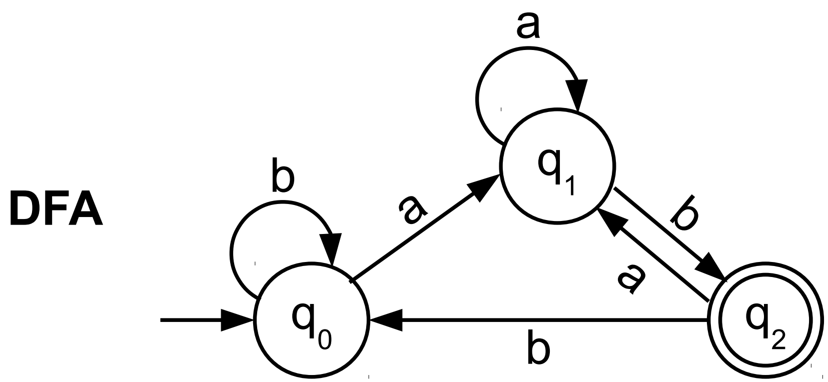
- Types d'automates
  - Automates à **états finis**
    - **FSA** : **F**inite **S**tate **A**utomata
  - Automates à états finis **déterministes**
    - **DFA** : **D**eterminist **F**inite state **A**utomata
  - Automates à états finis **non-déterministes**
    - **NFA** : **N**on-determinist **F**inite state **A**utomata
  - Automates à états finis **non-déterministes** avec possiblité de transitions ε (mot vide)
    - **ε-NFA** : **N**on-determinist **F**inite state **A**utomata with **ε** transitions
- Tous peuvent être représentés à l'aide des diagrammes de transition, tables de transition, notations formelles

- **Déterminisme** : requiert que l'automate soit dans un état **unique** (« déterminé ») après avoir lu n symboles
- Automates **non-déterministes** (**NFA**)
  - Non-déterminisme : l'automate peut-être dans **plusieurs états**, « **simultanément** » après avoir lu n symboles
    - **Attention** : un automate non-déterministe n'implique pas que l'on ne « sache » pas dans quels états il est (au contraire)
  - Particulièrement utile lorsque l'on ne sait pas « à l'avance » quel état final l'automate atteindra (**hypothèses**)
    - Facilité de représentation / programmation
      - GPS : une voiture se dirige vers Paris, mais on ne connaît pas la destination finale (Orléans, Strasbourg)
      - Prédiction SMS : un utilisateur tape le début d'un mot, mais plusieurs hypothèses sont possibles pour le mot qu'il veut écrire

- **Fonction de transition** d'un **NFA**
  - A partir d'un état et d'un symbole : **plusieurs** états possibles
    - Notation : $\delta(q_i, a) = R \subseteq Q$ (« signature » $\delta : Q \times \Sigma \rightarrow Q^*$)
      - Par ex. $\delta(q_i, a) = \{ q_j, q_k, q_l \}$
    - Peut aussi être l'ensemble vide : $\delta(q_i, a) = \varnothing$
- Différence de représentations **DFA / NFA**

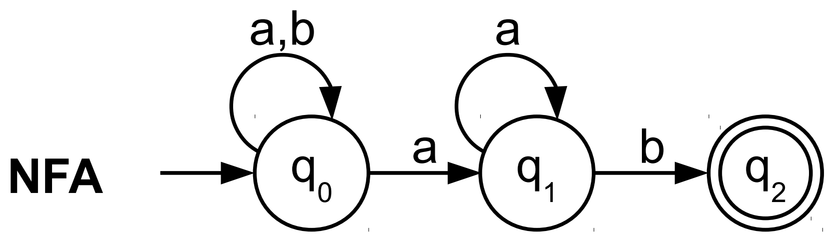
	DFA	NFA
Diagramme de transitions	Pour chaque paire (état, symbole), au maximum un arc sortant	Possibilité de plusieurs arcs sortant par paire (état, symboles)
Table de transitions	Les cases de la table contiennent un état ou Ø	Les cases de la table contiennent un ensemble d'états
Notation formelle	La fonction de transition renvoie au maximum un état	La fonction de transition peut renvoyer plusieurs états

- Différence de représentations **DFA / NFA** (suite)
  - Soit $\Sigma = \{ a, b \}$, comment représenter un automate qui reconnaît tous les mots se terminant par ab ?



⟺

		a	b
→	$\mathbf{q_0}$	q_1	q_0
	$\mathbf{q_1}$	q_1	q_2
*	$\mathbf{q_2}$	q_1	q_0



⟺

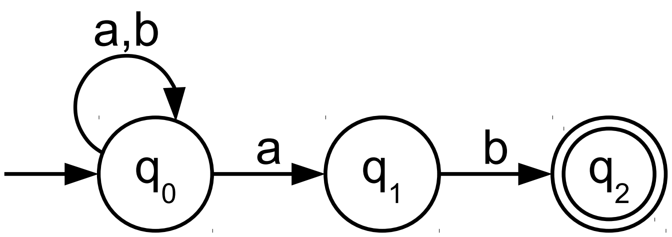
		a	b
→	$\mathbf{q_0}$	$\{ q_0, q_1 \}$	$\{ q_0 \}$
	$\mathbf{q_1}$	{ a }	$\{ q_2 \}$
*	$\mathbf{q_2}$	Ø	Ø

- **Fonction de transition étendue** d'un **NFA**
  - A partir d'un état et d'un mot, plusieurs états possibles
    - Fonction qui prend un mot en entrée et utilise δ du NFA
    - Notation ($\alpha \in \Sigma^*$) : $\overline{\delta}(q_i, \alpha) = R \subseteq Q$ (« signature » $\delta : Q \times \Sigma^* \rightarrow Q^*$)
    - De même, définition **récursive** par **décomposition** du mot w :
      - Si $w = \varepsilon$ alors $\overline{\delta}(q_i, \varepsilon) = \{ q_i \}$
      - Si $w = x.a$ tels que $x \in \Sigma^*$, $a \in \Sigma$ et $R = \overline{\delta}(q_i, x)$ alors

        $\overline{\delta}(q_i, w) = \cup_{r \in R} \delta(r, a)$

        ou (équivalent) : $\overline{\delta}(q_i, w) = \cup_{qj \in \overline{\delta}(qi, x)} \delta(q_j, a)$
- **Langage reconnu** par un NFA
  - Mots w tels que $\overline{\delta}(q_0, w)$ contienne **au moins un état final**
  - $L(A) = \{ w \in \Sigma^* \mid \overline{\delta}(q_0, w) \cap F \neq \varnothing \}$

- Par ex. soit l'automate ci-dessous et w = baababbbab



		a	b
→	q_0	$\{ q_0, q_1 \}$	$\{ q_0 \}$
	q_1	Ø	$\{ q_2 \}$
*	q_2	Ø	Ø

$\overline{\delta}(q_0, b) = \{q_0\}$

$\overline{\delta}(q_0, ba) = \cup_{p \in \{q0\}} \delta(p, a) = \delta(q_0, a) = \{q_0, q_1\}$

$\overline{\delta}(q_0, baa) = \cup_{p \in \{q0, q1\}} \delta(p, a) = \delta(q_0, a) \cup \delta(q_1, a) = \{q_0, q_1\}$

$\overline{\delta}(q_0, baab) = \cup_{p \in \{q0, q1\}} \delta(p, b) = \delta(q_0, b) \cup \delta(q_1, b) = \{q_0, q_2\}$

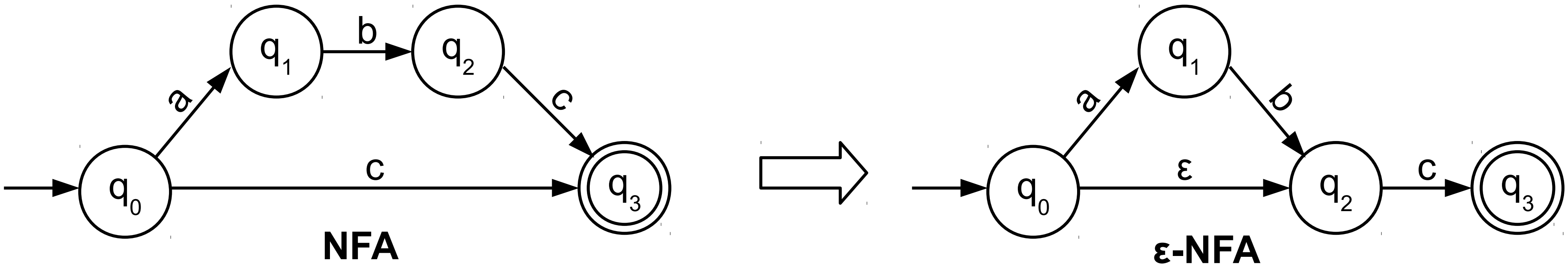
$\overline{\delta}(q_0, baaba) = \cup_{p \in \{q0, q2\}} \delta(p, a) = \delta(q_0, a) \cup \delta(q_2, a) = \{q_0, q_1\}$

...

$\overline{\delta}(q_0, baababbbab) = \{q_0, q_2\}$

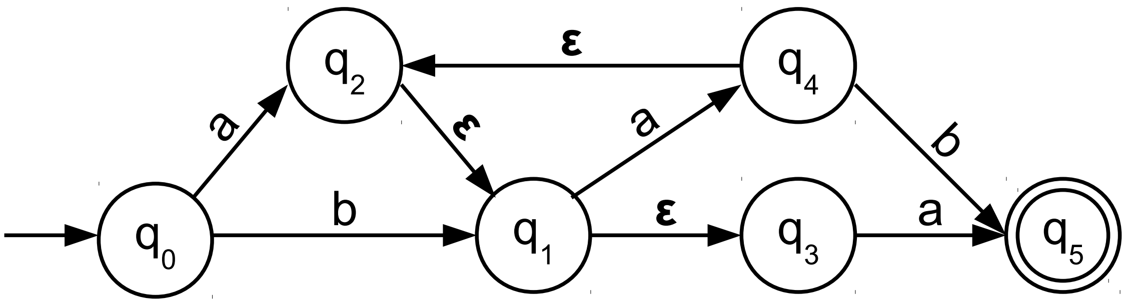
donc $\overline{\delta}(q_0, baababbbab) \cap F = \{q_2\} \neq Ø$ et $w \in L(A)$

- Automates avec transitions ε (ε-**NFA**)
  - **Transition ε** : transitions sur le mot vide → aucun symbole
    - La transition se réalise systématiquement
  - Facilité supplémentaire de représentation / programmation
    - Lorsque des parties sont « optionnelles »
  - Tout automate qui comporte une transition ε est **non-déterministe** (peut-être dans plusieurs états simultanément)
- Par ex., les mots abc ou c :



- **Fonction de transition** d'un **ε-NFA**
  - Identique à celle du **NFA**, mais en tenant compte des transitions ε comme élément pour réaliser une transition
    - Le mot-vide ε n'est pas nécessairement un élément de l'alphabet Σ
    - Notation : $\delta(q_i, a) = R \subseteq Q$ (« signature » $\delta : Q \times \Sigma \cup \boldsymbol{\{\varepsilon\}} \rightarrow Q^*$)
- **Fermeture** (transitive) par **transition ε**
  - Fermeture transitive (clôture, algèbre) ≃ éléments que l'on peut atteindre en utilisant un opérateur donné récursivement
  - Définition d'une fonction récursive « **epsClos** » :
    - Pour tout $q_i \in Q$, $\boldsymbol{q_i \in \mathrm{epsClos}(q_i)}$
    - Pour tout $q_i \in Q$, si $\delta(q_i, \varepsilon) = R \subseteq Q$ alors $\boldsymbol{R \subset \mathrm{epsClos}(q_i)}$
  - On « suit » **toutes les transitions ε possibles**, récursivement

- Par ex. :



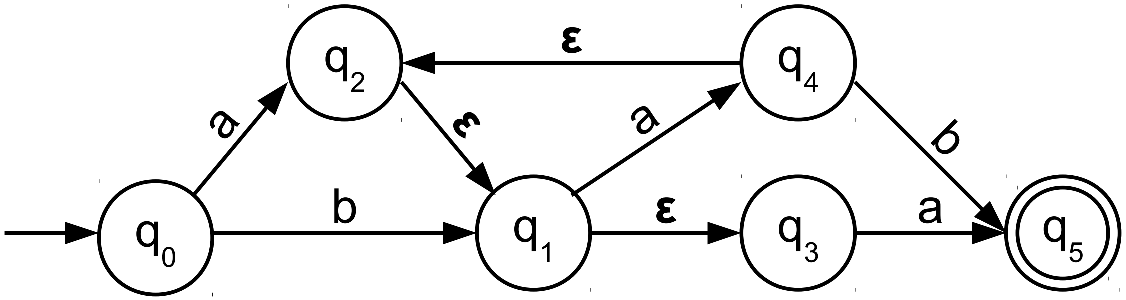
  - epsilonClosure(q_0) = {q_0}
  - epsilonClosure(q_1) = {q_1, $\mathbf{q_3}$}
  - epsilonClosure(q_2) = {q_2, $\mathbf{q_1}$, $\mathbf{q_3}$}
  - epsilonClosure(q_3) = {q_3}
  - epsilonClosure(q_4) = {q_4, q_2, $\mathbf{q_1}$, $\mathbf{q_3}$}
  - epsilonClosure(q_5) = {q_5}

- **Fonction de transition étendue** d'un **ε-NFA**
  - Même approche que pour un NFA
    - Fonction qui prend un mot en entrée et utilise δ du ε-NFA
    - Notation ($\alpha \in \Sigma^*$) : $\bar{\delta}(q_i, \alpha) = R \subseteq Q$ (« signature » $\delta : Q \times \Sigma^* \rightarrow Q^*$)
    - Transitions tiennent compte des transitions ε avec epsClos
    - De même, définition **récursive** par **décomposition** du mot w :
      - Si $w = \varepsilon$ alors $\bar{\delta}(q_i, w) = \text{epsClos}(q_i)$
      - Si $w = x.a$ tels que $x \in \Sigma^*$, $a \in \Sigma$ et $R = \bar{\delta}(q_i, x)$ alors

        $\bar{\delta}(q_i, w) = \cup_{r \in R} \cup_{s \in \delta(r, a)} \text{epsClos}(s)$

        ou (équivalent) : $\bar{\delta}(q_i, w) = \cup_{qj \in \bar{\delta}(qi, x)} \cup_{qk \in \delta(qj, a)} \text{epsClos}(q_k)$
- **Langage reconnu** par un ε-NFA (comme pour NFA)
  - $L(A) = \{ w \in \Sigma^* \mid \bar{\delta}(q_0, w) \cap F \neq \emptyset \}$

- Par ex. :



  - $\overline{\delta}(q_0, a) = \{q_2, q_1, q_3\}$
  - $\overline{\delta}(q_0, b) = \{q_1, q_3\}$
  - $\overline{\delta}(q_0, aa) = \{q_5, q_4, q_2, q_1, q_3\} = \overline{\delta}(q_0, ba) = \overline{\delta}(q_0, aaa)$
  - $\overline{\delta}(q_0, bb) = \varnothing$
  - $\overline{\delta}(q_0, bab) = \{q_5\} = \overline{\delta}(q_0, baab) = \overline{\delta}(q_0, aab) = \overline{\delta}(q_0, aaab)$

Plan

- Représentation des automates (FSA)
- Définition formelle (DFA)
- Déterminisme du FSA (DFA / NFA / ε-NFA)
- Équivalence DFA / NFA / ε-NFA

- Les automates **reconnaissent** des langages
  - Représentation de l'automate plus ou moins « dense »
  - Distinction entre **DFA**, **NFA** et **NFA**
    - Fonctions de transitions
    - Nombre d'états atteints depuis un état en consommant un symbole
  - Le **langage reconnu** par un automate ne dépend pas des états de l'automate
    - Plusieurs automates possibles pour un même langage
- **Équivalence** des automates (langages reconnus)
  - Automates **DFA** et **NFA**
  - Automates **NFA** et **ε-NFA**

- **Équivalence DFA / NFA**
  - Tout **DFA** peut-être considéré comme un **NFA**
  - Soit un **NFA**, N = $(Q_N, \Sigma_N, \delta_N, q_{N0}, F_N)$
    - Il existe un **DFA**, D = $(Q_D, \Sigma_D, \delta_D, q_{D0}, F_D)$ tel que L(D) = L(N)
      - Même alphabet ($\Sigma_D = \Sigma_N$), même état initial ($q_{D0} = q_{N0}$)
      - On **construit** Q_D comme l'ensemble des sous-ensemble de Q_N
        - Par ex. si $Q_N = \{q_0, q_1\}$ alors $Q_D = \{\{q_0\}, \{q_1\}, \{q_0 q_1\}\}$
      - États finaux F_D : ceux qui sont **construits** avec un état final de F_N

        $F_D = \{q_{Di} \in Q_D \mid \{q_{Ni} \in q_{Di}\} \cap F_N \neq \emptyset\}$
      - Fonction de transition δ_D de D qui, à partir d'un état de Q_D comme **construction** de Q_N renvoie vers une autre construction de Q_N :

        $\delta_D(q_{Di}, a) = \{ \cup_{q_{Nj} \in q_{Di}} \delta_N(q_{Nj}, a) \}$

- **Équivalence DFA / NFA** (suite)
  - Preuve sur la longueur du mot w = x.a
    - Supposons que : $\bar{\delta}_D(q_{D0}, x) = \bar{\delta}_N(q_{N0}, x) = Q$
    - Par définition du NFA :
      - $\bar{\delta}_N(q_{N0}, x.a) = \cup_{qNj \in \bar{\delta}(qN0,\, x)} \delta(q_{Nj}, a) = \cup_{qNj \in Q} \delta_N(q_{Nj}, a)$
    - Et par construction du DFA :
      - $\delta_D(q_{Di}, a) = \{ \cup_{qNj \in qDi} \delta_N(q_{Nj}, a) \}$
      - $\bar{\delta}_D(q_{D0}, x.a) = \delta_D(\bar{\delta}_D(q_{D0}, x), a) = \delta_D(Q, a) = \{ \cup_{qNj \in Q} \delta_N(q_{Nj}, a) \}$
    - Alors $\bar{\delta}_N(q_{N0}, x.a) = \cup_{qNj \in Q} \delta_N(q_{Nj}, a) = \bar{\delta}_D(q_{D0}, x.a)$
  - **Récursivité** : le DFA et le NFA aboutissent aux mêmes états, mais dans le cas du DFA c'est « par construction »
  - Pour tout NFA, il est possible de **construire** un DFA qui reconnaîtra le **même langage**.

- **Équivalence NFA / ε-NFA**
  - Tout **NFA** peut-être considéré comme un **ε-NFA**
  - Soit un **ε-NFA**, E = $(Q_E, \Sigma_E, \delta_E, q_{E0}, F_E)$
    - Il existe un **NFA**, N = $(Q_N, \Sigma_N, \delta_N, q_{N0}, F_N)$ tel que L(N) = L(E)
      - Même alphabet ($\Sigma_N = \Sigma_E$)
      - On **construit** Q_N comme l'ensemble des sous-ensemble de Q_E
      - État initial **construit** $q_{N0} = \{epsClos(q_{E0})\}$
      - États finaux F_N : **construits** avec un état final de F_E

        $F_N = \{q_{Ni} \in Q_N \mid \{q_{Ei} \in q_{Ni}\} \cap F_E \neq \emptyset\}$
      - Fonction de transition δ_N de N qui, à partir d'un état de Q_N comme **construction** de Q_E renvoie vers une autre construction de Q_E :

        $\delta_N(q_{Ni}, a) = \{ \cup_{qEj \in qNi} \cup_{qNk \in \delta E(qNi,\, a)} epsClos(q_{Nk}) \}$