Langages et Automates : LA3
Partie 6 : Minimisation

Minimisation

Dans ce chapitre, on va montrer l'existence d'un unique automate ayant un nombre minimum d'état pour un langage rationnel fixé.
On va montrer comment définir cet automate, soit à partir d'une descrption énumérative ou d'une expression rationnelle, soit en partant d'un automate et en le minimisant pour obtenir l'automate minimal.

Résiduels

Définition

Pour un mot u et un langage L, on définit le résiduel de u par rapport à L

$$u^{-1}L = \{v \in \Sigma^*, uv \in L\}$$

On prend tous les mots de L qui commencent par u et on tronque u.

On prend tous les mots v qu'on peut contaténer à u pour tomber dans L.

Résiduels - Exemples

- $L =$ mots qui commencent par aa.
  Si u commence par un b ou si u commence par un ab : $u^{-1}L = \emptyset$
  Si u commence par un aa : $u^{-1}L =$ tous les mots
  Si $u = a$: $u^{-1}L =$ mots commençant par un a.
  Si $u = \varepsilon$: $u^{-1}L =$ mots commençant par un $aa = L$.
- $L = \{u \in \{a,b\}^* \quad |u|_a = 0 \mod 3\}$.
  Si $|u|_a = 0 \mod 3$, alors $u^{-1}L = \{u \in \{a,b\}^* \quad |u|_a = 0 \mod 3\} = L$
  Si $|u|_a = 1 \mod 3$, alors $u^{-1}L = \{u \in \{a,b\}^* \quad |u|_a = 2 \mod 3\} = L_2$
  Si $|u|_a = 2 \mod 3$, alors $u^{-1}L = \{u \in \{a,b\}^* \quad |u|_a = 1 \mod 3\} = L_1$
- $L = \{u$ tels que u contient le facteur $bb\}$
  Si $u \in L$, $u^{-1}L = \Sigma^*$
  Si $u \notin L$ et u finit par un b, $u^{-1}L = L \cup b\Sigma^*$
  Si $u \notin L$ et u ne finit pas par un b, $u^{-1}L = L$

Résiduels - Propriétés

Quelques propriétés importantes

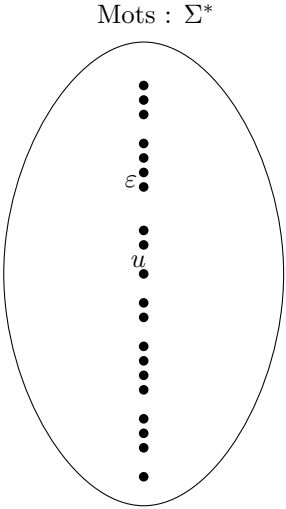
- Pour tout langage L, $\varepsilon^{-1}L = \mathrm{L}$
- $u \in L$ si et seulement si $u^{-1}L$ contient le mot vide.
- Si $u \in L$ et $u^{-1}.L = v^{-1}L$, alors $v \in L$.
- $(u.a)^{-1}L = a^{-1}(u^{-1}L)$.
- Si $u^{-1}.L = v^{-1}L$, alors $(u.a)^{-1}L = (v.a)^{-1}L$ (pour toute lettre a).

Résiduels - Propriétés 2

On peut se servir des identités suivantes pour calculer différents résiduels (u est un mot, a est une lettre)

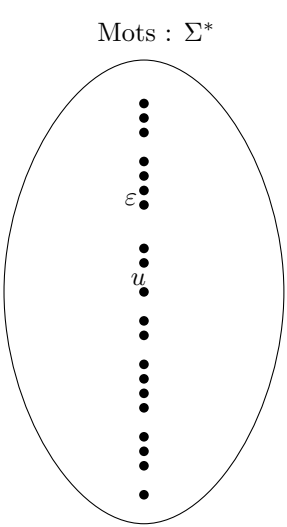
- $(uv)^{-1}L = v^{-1}(u^{-1}L)$
- $u^{-1}(L_1 \cup L_2) = (u^{-1}L_1) \cup (u^{-1}L_2)$
- $a^{-1}(L_1 \cdot L_2) = \begin{cases} (a^{-1}L_1) \cdot L_2 & \text{si } \varepsilon \notin L_1 \\ (a^{-1}L_1) \cdot L_2 \cup (a^{-1}L_2) & \text{si } \varepsilon \in L_1 \end{cases}$
- $u^{-1}(L^*) = (u^{-1}L) \cdot L^*$

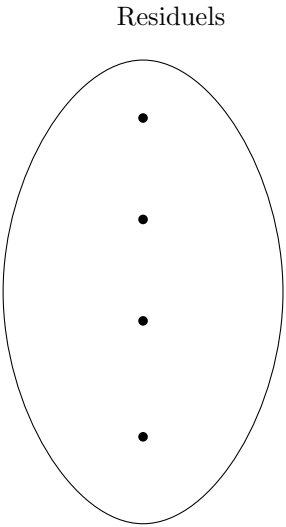
Un dessin



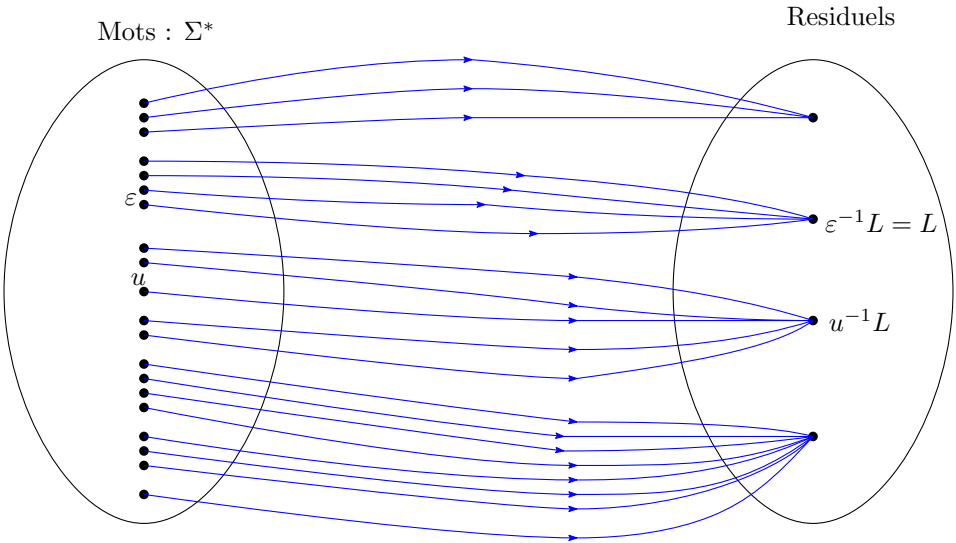
Residuels

Un dessin

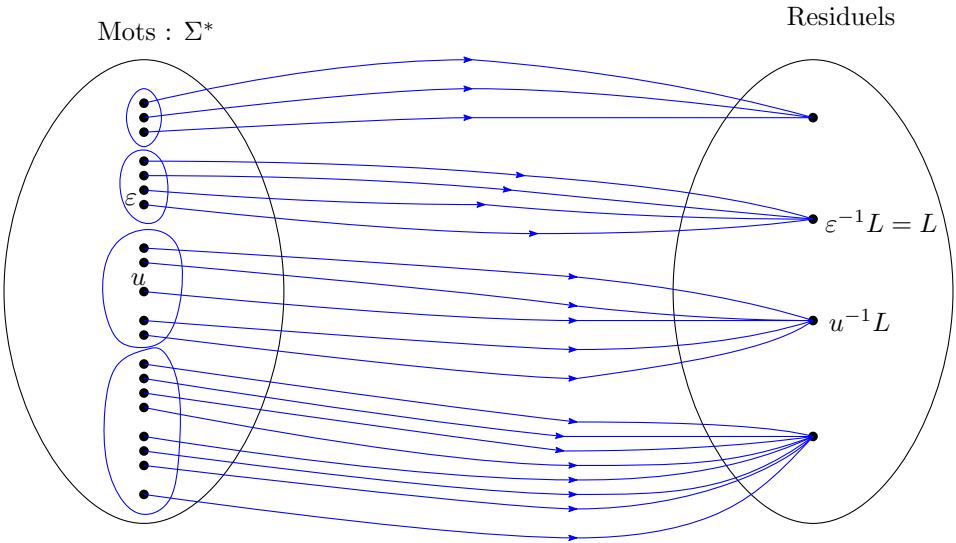




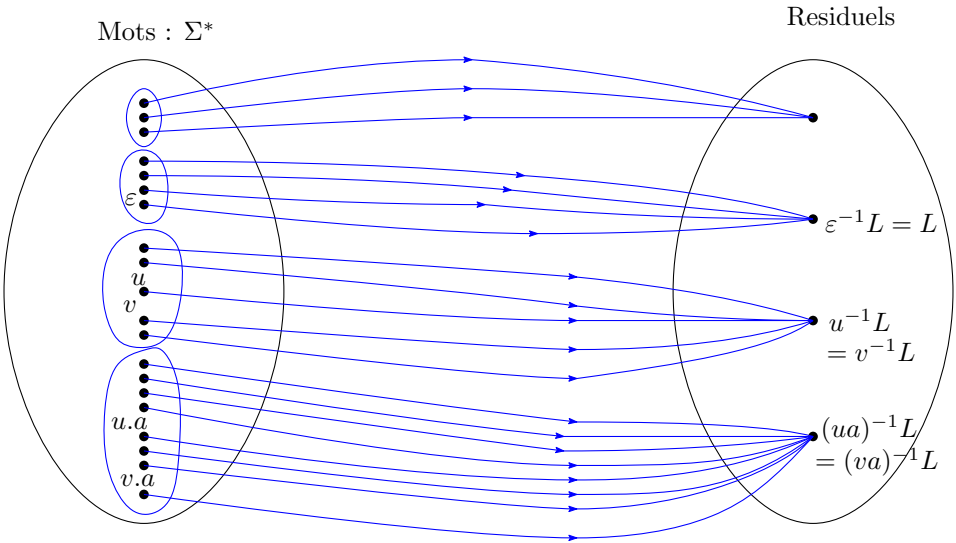
Un dessin



Un dessin



Un dessin


Mots : Σ^*
Residuels
ε
u
v
$u.a$
$v.a$
$\varepsilon^{-1}L = L$
$u^{-1}L$
$= v^{-1}L$
$(ua)^{-1}L$
$= (va)^{-1}L$

Définition

Soit Σ un alphabet et L un langage sur Σ.

- *On définit une relation d'équivalence $\sim_L$ sur Σ^* par :*

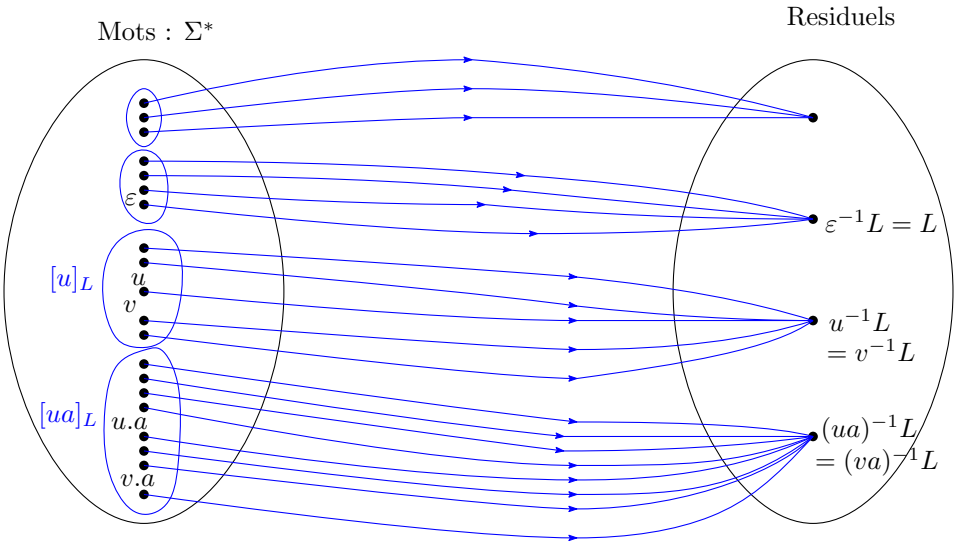
$$u \sim_L v \text{ si et seulement si } u^{-1}L = v^{-1}L.$$

- *Pour un mot u, on note $[u]_L$ la classe d'équivalence de u pour cette relation, c'est à dire l'ensemble des mots v tels que $u \sim_L v$.*
- *Deux mots u et v sont dit séparés par z si exactement un seul des mots $u.z$ et $v.z$ appartient à L.*

Pour condenser les notations et définitions introduites, on peut écrire donc

$$\begin{aligned} [u]_L = [v]_L &\Leftrightarrow u \sim_L v \\ &\Leftrightarrow u^{-1}L = v^{-1}L \\ &\Leftrightarrow u \text{ et } v \text{ ne sont séparés par aucun mot} \end{aligned}$$

Les Classes d'équivalence



Exemples :

1. $L = \{u \in \{a,b\}^* \ \ |u|_a = 0 \mod 3\}$. Deja vu, 3 classes : $[\varepsilon], [a], [aa]$.
2. $L = \{u \in \{a,b\}^* \ \ u$ contient le facteur $bb\ \}$. L admet trois classes d'équivalence :
   - $[bb] = L$ (résiduel Σ^*)
   - $[b] = \{$ mots pas dans L finissant par un $b\}$ (résiduel $= b\Sigma^* \cup L$)
   - $[\varepsilon] = \{$ mots pas dans L ne finissant pas par un $b\}$ (résiduel L)
3. $\{a^n b^n,\ n \in \mathbb{N}\}$
   Pour $p \neq q$, a^p et a^q sont séparés par b^p. Ainsi tous les mots a^n, $n \in \mathbb{N}$ sont dans des classes différentes Il y a donc ici une infinité de classes distinctes.

Théoreme (Myhill - Nérode)

Soit L un langage.

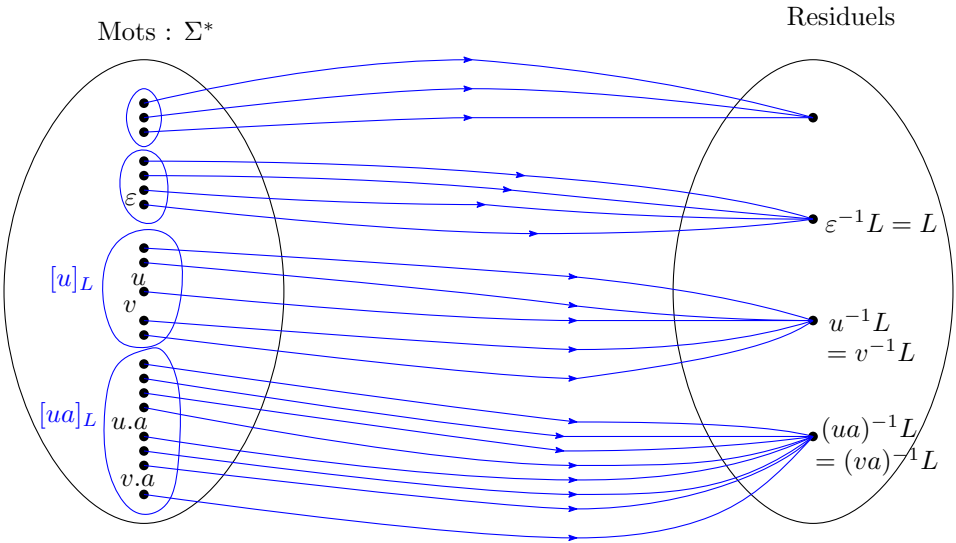
- *L est rationnel si et seulement si la relation $\sim_L$ a un nombre fini de classes.*
- *Si c'est le cas, soit p ce nombre classes. Alors tout AFD complet reconnaissant L a au moins p états et il existe un unique (au nom des sommets près) AFD complet reconnaissant L ayant exactement p états, on l'appelle l'automate minimal du langage L.*

Avant de passer à sa preuve, remarquons que ce théorème offre donc une nouvelle caractérisation des langages rationnels (nombre fini de classes).

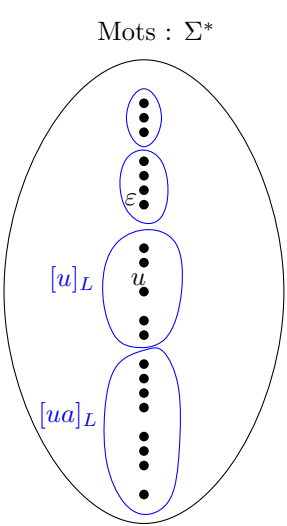
Exercice

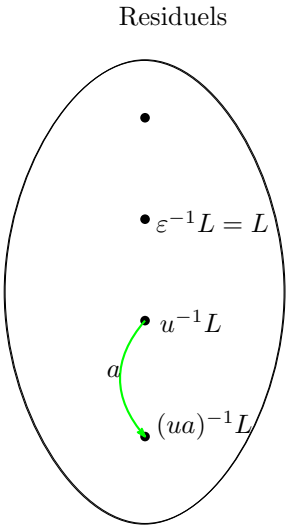
Utiliser ce théorème pour montrer que $\{a^n b^n,\ n \in \mathbb{N}\}$ n'est pas rationnel.
De même pour $\{a^{2^n},\ n \in \mathbb{N}\}$.

Classes d'équivalence



Classes d'équivalence





Construction d'un automate canonique

Supposons que $\sim_L$ a un nombre fini résiduels différents (ou encore : un nb fini de classes d'équivalence)

On va construire un automate par :

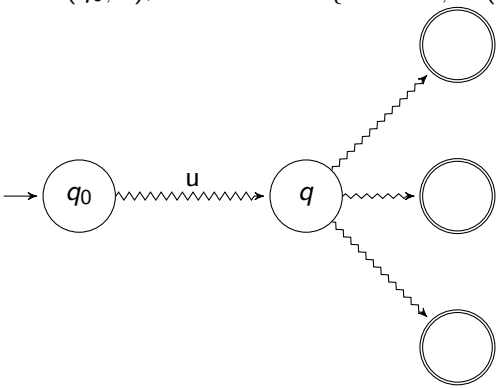
- Etats : les résiduels (en nombre fini)
- Etat initial : $\varepsilon^{-1}L$
- Transitions : $\delta(u^{-1}L, a) = (ua)^{-1}L$
  (bien définies car $u^{-1}.L = v^{-1}L \Rightarrow (u.a)^{-1}L = (v.a)^{-1}L$)
- Etats Finaux : l'ensemble des résiduels des mots de L (ou encore : l'ensemble des résiduels qui contiennent ε)

La fonction de transition vérifie donc :

$$\delta^*(e^{-1}L, u) = u^{-1}L$$

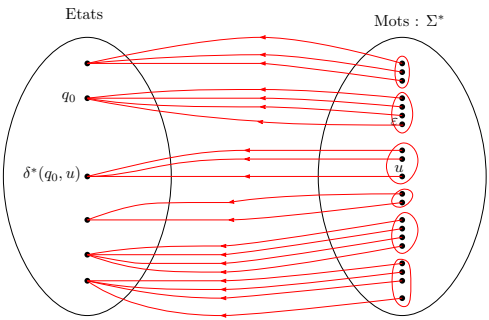
L'ensemble des mots reconnus par A est alors bien le langage L car $u^{-1}L$ est un par définition état final si u est dans L.

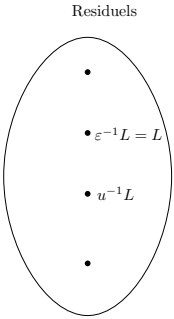
Dans le cas où le langage L est décrit par un automate $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ et si $q = \delta^*(q_0, u)$, alors $u^{-1}L = \{w \in \Sigma^*, \delta^*(q, w) \in F\}$.



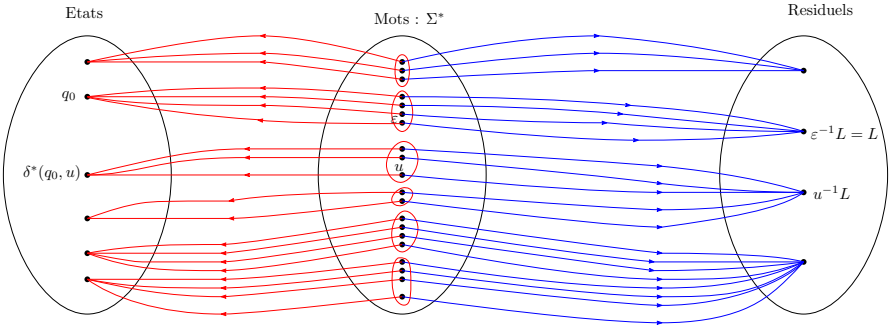
En conséquence si u et v sont deux mots tels que $\delta^*(q_0, u) = \delta^*(q_0, v)$, alors ils sont équivalents.

Classes d'équivalence

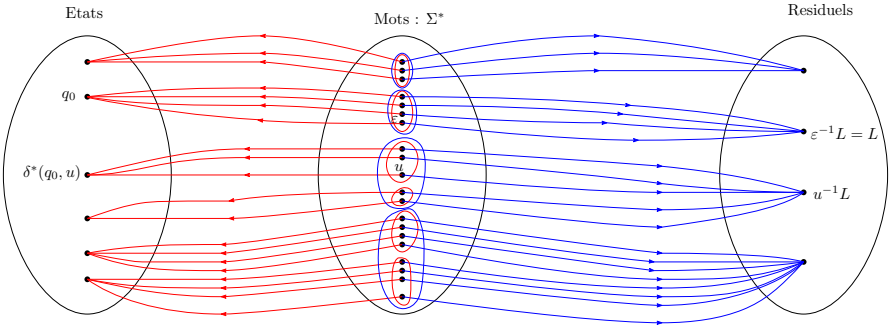




Classes d'équivalence



Classes d'équivalence



Preuve du Théorème de Myhill Nérode

Supposons maintenant L rationnel et donc reconnaissable, et soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un automate déterministe et complet tel que $L(A) = L$.

Important : si $\delta^*(q_0, u) = \delta^*(q_0, v)$, alors $u \sim_L v$.

Cela signifie que si on définit une relation d'équivalence sur les mots par $u \sim_A v$ si $\delta^*(q_0, u) = \delta^*(q_0, v)$, alors les classes d'équivalence pour cette relation sont toutes incluses dans une classe de la relation $\sim_L$ (voir dessin précédent)

Le nombre de classes de $\sim_L$ est donc inférieur ou égal aux nombre de classes de $\sim_A$ qui est lui meme inférieur ou égal au nombre d'états de l'automate (égal si tous les états sont accessibles)

Si le nombre d'état est précisemment égal au nombre de résiduels, il est alors facile de voir que l'automate est alors identique à l'automate canonique construit précédemment, ce qui prouve bien l'unicité de l'automate minimal.

Résiduels - Calcul

Le théorème précédent implique que si on sait calculer tous les résiduels possibles, alors on pourra construire l'automate minimal.

Peut on le faire à partir d'une expression rationnelle ?

Si E expression rationnelle et u un mot, est-ce que l'expression $u^{-1}E$ a un sens ?

Dans le cas des langages reconnaissables par un AFD, on la la propriété importante suivante : pour un mot u, si $q = \delta^*(q_0, u)$, alors

$$u^{-1}L = \{w \in \Sigma^*\,,\, \delta^*(q, w) \in F\}$$

Cela prouve en particulier que tout résiduel d'un langage reconnaissable est aussi reconnaissable (il suffit de changer l'état initial en q).

En conséquence, si E est une expression rationnelle décrivant un langage L, on définit alors $u^{-1}E$ l'expression rationnelle correspondant au langage $u^{-1}L$.

Résiduels - Calcul

Toutes les identitiés vues sur les langages sont se réécrivent donc pour les E.R.

- $(uv)^{-1}E = v^{-1}(u^{-1}E)$
- $u^{-1}(E_1 + E_2) = (u^{-1}E_1) + (u^{-1}E_2)$
- $u^{-1}(E_1 \cdot E_2) =$
  $$\begin{cases} (u^{-1}E_1) \cdot E_2 & \text{si } \varepsilon \text{ n'appartient pas au langage décrit par } E_1 \\ (u^{-1}E_1) \cdot E_2 + u^{-1}E_2 & \text{si } \varepsilon \text{ appartient au langage décrit par } E_1 \end{cases}$$
- $u^{-1}(E^*) = (u^{-1}L) \cdot E^*$

Un cas particulier de la première indentité est celui ou v est un lettre :

$\Rightarrow$ on peut calculer les résiduels des mots de longueur k à partir de ceux des mots de longueur $k-1$.

On peut calculer tous les résiduels en commençant par les mots de longueur 0, puis 1, puis 2, etc...

Résiduels - Exemple de calcul

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

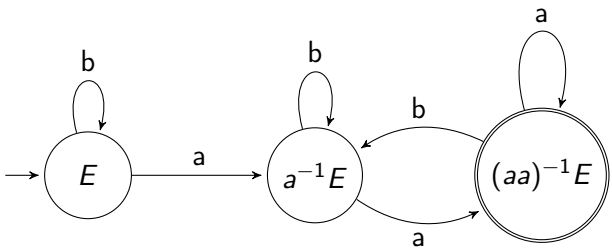
$$\begin{aligned} a^{-1}E &= a^{-1}((a+b)^*).a(a+b)^*a + a^{-1}(a(a+b)^*a) \\ &= a^{-1}((a+b)).(a+b)^*a(a+b)^*a + (a+b)^*a \\ &= (a+b)^*a(a+b)^*a + (a+b)^*a \\ &= E + (a+b)^*a \\ &= (a+b)^*a \end{aligned}$$

$$\begin{aligned} (ab)^{-1}E &= b^{-1}(a^{-1}L) \\ &= b^{-1}((a+b)^*a) \\ &= b^{-1}((a+b)^*)a + b^{-1}a \\ &= (a+b)^*a \end{aligned}$$

Automate Minimal par la méthode des résidus

Exemple :

$$
\begin{aligned}
\varepsilon^{-1}E &= E = (a+b)^*a(a+b)^*a \\
a^{-1}E &= (a+b)^*a \\
b^{-1}E &= (a+b)^*a(a+b)^*a = \boxed{E} \\
(aa)^{-1}E = a^{-1}(a^{-1}E) &= (a+b)^*a(a+b)^*a + (a+b)^*a + \varepsilon \\
(ab)^{-1}E = b^{-1}(a^{-1}E) &= (a+b)^*a(a+b)^*a + (a+b)^*a = \boxed{a^{-1}E} \\
(aaa)^{-1}E = a^{-1}((aa)^{-1}E) &= (a+b)^*a(a+b)^*a + (a+b)^*a + \varepsilon = \boxed{(aa)^{-1}E} \\
(aab)^{-1}E = b^{-1}((aa)^{-1}E) &= (a+b)^*a(a+b)^*a + (a+b)^*a = \boxed{a^{-1}E}
\end{aligned}
$$

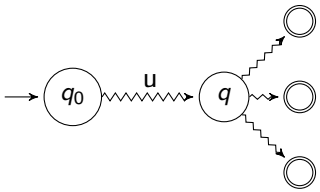


Automate Minimal par Partitionnement

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un AFD. Pour tout état $q \in Q$, on définit

$$L_q = \{w \in \Sigma^* \text{ tels que } \delta^*(q, w) \in F\}.$$

Si q est un état accessible par le mot u , alors il s'agit bien du résidu associé à u :



Si $\delta(q_0, u) = q$ alors $L_q = u^{-1}L$

C'est pourquoi on supposera par la suite que tous les états de l'automates sont accessibles, c'est à dire que pour tout $q \in Q$, il existe $u \in \Sigma^*$ tel que $\delta(q_0, u) = q$ (Il est évident qu'un état qui n'est pas accessible ne "sert à rien " dans l'automate : on peut le supprimer sans changer le langage reconnu).
Si tous les états sont accessible on a donc bien qu'à chaque état correspond un résidu du langage qu'il reconnait.

Automate Minimal par Partitionnement

Soit $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$ un AFD. Pour tout état $q \in Q$, on définit

$$L_q = \{w \in \Sigma^* \text{ tels que } \delta^*(q, w) \in F\}.$$

Définition

On définit une relation d'équivalence $\equiv$ entre états de l'automate par :

$$q \equiv q' \text{ si seulement si } L_q = L_{q'}$$

(La classe d'équivalence d'un état q est notée $[q]_{\mathcal{A}}$)

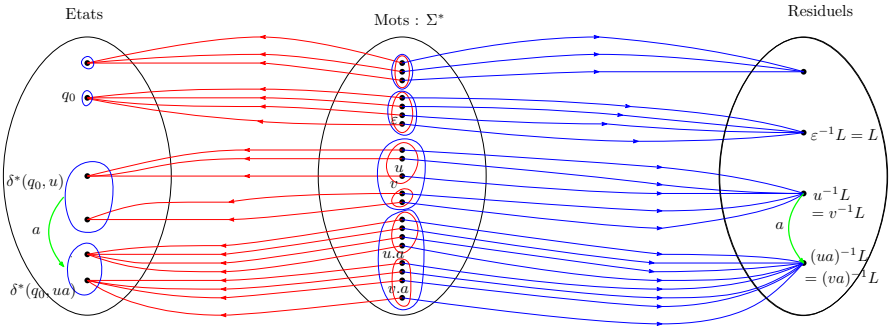
(Attention, ceci est une relation d'équivalence sur les ETATS, a la différence de $\sim_L$ qui était précédemment une relation d'équivalence sur les mots dans Σ^*).

Proposition

Si L est le langage reconnu par l'automate $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$:

$$(u \sim_L v) \Leftrightarrow \left(u^{-1}L = v^{-1}L\right) \Leftrightarrow \left(\delta^*(q_0, u) \equiv \delta^*(q_0, v)\right).$$

Automate Minimal par Partitionnement



On construit un automate en "fusionnant" les états équivalents.

Pour cela, il faut s'assurer que les transitions ne vont pas poser de problème.
Ceci est garanti par le fait que :

$$q \equiv q' \text{ implique } \forall a \in \Sigma\ ,\ \delta(q, a) \equiv \delta(q', a)$$

Automate Minimal par Partitionnement

Supposons que $\mathcal{A}$ est un AFD accessible et complet.
Formellement, ces fusions d'états équivalents définissent $\mathcal{A}' = (\Sigma', Q', q_0', F', \delta')$.

- états : $Q' =$ l'ensemble des classes d'équivalence $[q]_{\mathcal{A}}$ de $\equiv$.
- état initial : $q_0' = [q_0]_{\mathcal{A}}$
- états finaux : $F' = \{[q]_{\mathcal{A}},\ q \in F\}$
- transitions : $\delta'([q]_{\mathcal{A}}, a) = [\delta(q, a)]_{\mathcal{A}}$ pour tout état q et lettre a.

Il n'y a plus d'états équivalents, tous les états sont séparés, et il y a donc autant d'états que de résidus du langage.
Autrement dit, cet automate est exactement l'automate minimal construit précédemment.

Algorithme de Moore pour le calcul des classes d'équivalence

Pour pouvoir trouver l'automate minimal a partir d'un automate quelconque, il faut donc etre capable d'identifier les états équivalents.
On rappelle qu'on a défini $L_q = \{w \in \Sigma^* \text{ tels que } \delta^*(q, w) \in F\}$ et

$$q \equiv q' \text{ si seulement si } L_q = L_{q'}$$

ou de façon équivalente :

$$q \equiv q' \text{ si seulement si aucun mot ne sépare } q \text{ et } q'$$

Au lieu de calculer les classes directement, on va faire progressivement : on commence par voir si deux états sont séparés par des mots de longueur 1. Si ce n'est pas le cas, on cherche si ils le sont pour des mots de longueur 2, et ainsi de suite.

On va voir qu'il suffit de faire ça un nombre fini de fois pour être sur que deux états ne sont pas séparés par aucun mot.

Algorithme de Moore pour le calcul des classes d'équivalence

Notons Σ_k l'ensemble des mots de longueurs inférieure ou égale à k.

On définit la relation d'équivalence $\equiv_k$ par :

$$q \equiv_k q' \text{ si seulement si } L_q \cap \Sigma_k = L_{q'} \cap \Sigma_k$$

ou de façon équivalente :

$$q \equiv_k q' \text{ si seulement si aucun mot de longueur } \leq k \text{ ne sépare } q \text{ et } q'$$

Bien sur $q \equiv_k q'$ implique $q \equiv_{k-1} q'$ donc les classes d'équivalences $\equiv_k$ sont de plus en plus fines et on a :

$$q \equiv q' \text{ si seulement si } \forall k \in \mathbb{N}\ q \equiv_k q'$$

Algorithme de Moore pour le calcul des classes d'équivalence

On commence par calculer $\equiv_0$. Par définition $q \equiv_0 q'$ signifie que q et q' sont tous les deux acceptants ou tous les deux non acceptants. Autrement dit, σ_0 a deux classes : F et $Q \setminus F$.

On peut calculer les classes de $\equiv_k$ à partir des classes de $\equiv_{k-1}$ grâce à la propriété suivante

Proposition

$$q \equiv_k q' \text{ si seulement si } \begin{cases} q \equiv_{k-1} q' \\ \forall a \in \Sigma,\ \delta(q,a) \equiv_{k-1} \delta(q',a) \end{cases}$$

De plus, cela implique que si les classes d'équivalences sont toutes identiques pour $\equiv_{k-1}$ et $\equiv_k$, alors elles restent identiques pour tout $k' \geq k$ et sont donc exactement les classes de $\equiv$.

Proposition

$$q \equiv_k q' \text{ si seulement si } \begin{cases} q \equiv_{k-1} q' \\ \forall x \in \Sigma,\ \delta(q,x) \equiv_{k-1} \delta(q',x) \end{cases}$$

Preuve :
Par définition,

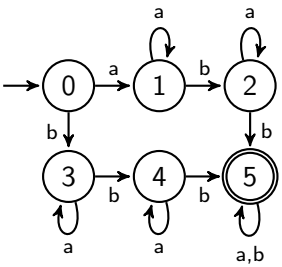
$$q \equiv_k q' \text{ si seulement si } \forall u \in \Sigma_k,\ \delta(q,u) \equiv_0 \delta(q',u)$$

Or,

$$\begin{array}{lll} & \forall u \in \Sigma_k \setminus \{\varepsilon\} & \delta(q,u) \equiv_0 \delta(q',u) \\ \text{si seulement si} & \forall x \in \Sigma\ \forall v \in \Sigma_{k-1} & \delta(q,xv) \equiv_0 \delta(q',xv) \\ \text{si seulement si} & \forall x \in \Sigma\ \forall v \in \Sigma_{k-1} & \delta(\delta(q,x),v) \equiv_0 \delta(\delta(q',x),v) \\ \text{si seulement si} & \forall x \in \Sigma & \delta(q,x) \equiv_{k-1} \delta(q',x) \end{array}$$

Ce qui prouve bien la Proposition.

Algorithme de Moore : Exemple

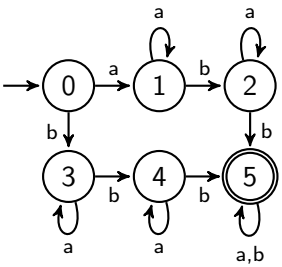


- classes de $\equiv_0$: $\{0,1,2,3,4\}$ $\{5\}$

Pour trouver les classes de $\equiv_1$, on applique la Proposition :

$$q \equiv_1 q' \text{ si seulement si } \begin{cases} q \equiv_0 q' \\ \forall a \in \Sigma,\ \delta(q,a) \equiv_0 \delta(q',a) \end{cases}$$

Algorithme de Moore : Exemple

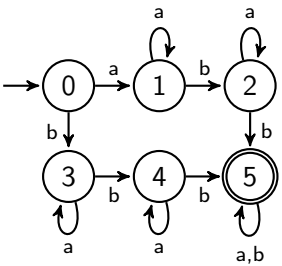


- classes de $\equiv_0$: $\{0,1,2,3,4\}$ $\{5\}$
- classes de $\equiv_1$: $\{0,1,3\}$ $\{2,4\}$ $\{5\}$

Pour trouver les classes de $\equiv_2$, on applique la Proposition :

$$q \equiv_2 q' \text{ si seulement si } \begin{cases} q \equiv_1 q' \\ \forall a \in \Sigma,\ \delta(q,a) \equiv_1 \delta(q',a) \end{cases}$$

Algorithme de Moore : Exemple

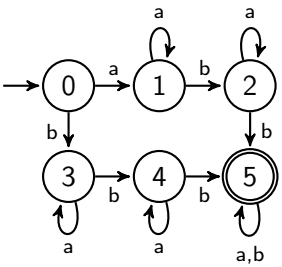


- classes de $\equiv_0$: $\{0,1,2,3,4\}$ $\{5\}$
- classes de $\equiv_1$: $\{0,1,3\}$ $\{2,4\}$ $\{5\}$
- classes de $\equiv_2$: $\{0\}$ $\{1,3\}$ $\{2,4\}$ $\{5\}$

Pour trouver les classes de $\equiv_3$, on applique la Proposition :

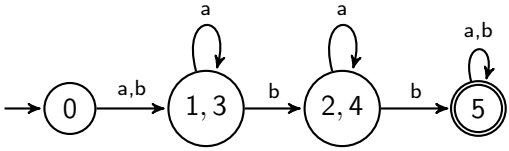
$$q \equiv_2 q' \text{ si seulement si } \begin{cases} q \equiv_1 q' \\ \forall a \in \Sigma,\ \delta(q,a) \equiv_1 \delta(q',a) \end{cases}$$

Algorithme de Moore : Exemple



- classes de $\equiv_0$: $\{0,1,2,3,4\}$ $\{5\}$
- classes de $\equiv_1$: $\{0,1,3\}$ $\{2,4\}$ $\{5\}$
- classes de $\equiv_2$: $\{0\}$ $\{1,3\}$ $\{2,4\}$ $\{5\}$
- classes de $\equiv_3$: $\{0\}$ $\{1,3\}$ $\{2,4\}$ $\{5\}$

On peut donc conclure que les classes de $\equiv$ sont bien : $\{0\}$ $\{1,3\}$ $\{2,4\}$ $\{5\}$.



Le Théorème de Brzozowski pour la minimisation

Pour L langage, on note $mir(L)$ le langage constitué du miroir des mots de L.
Si L est reconnu par l'automate $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$, alors $mir(L)$ est reconnu par l'automate (en général non déterministe) $mir(\mathcal{A}) = (\Sigma, Q, F, \{q_0\}, \delta_{mir})$ où

$$\delta_{mir}(q, a) = \{q' \in Q \text{ tels que } \delta(q, a) = q'\}$$

(on inverse le sens des transitions et on permute le role des états initiaux et finaux).
Brozowski a montré le résultat surprenant suivant qui fournit un autre moyen de calculer l'automate minimal.

Théoreme

Si L est un langage rationnel reconnu par l'automate $\mathcal{A}$, alors l'automate minimal $\mathcal{A}_L$ de L est donné par la formule suivante

$$\mathcal{A}_L = det(mir(det(mir(\mathcal{A}))))$$

Proposition

Soit L un langage et $\mathcal{B}$ un automate co-deterministe co-accessible (ie son miroir est deterministe et accessible) qui reconnait L.
Alors le determinisé $\mathcal{B}_{det}$ de $\mathcal{B}$ est l'automate minimal de L.

Preuve :

Etudions d'abord les résidus de l'automate $\mathcal{B}$. Pour tout état q de $\mathcal{B}$:

- il existe un mot u_q tel que $u_q \in L_q$ (car $\mathcal{B}$ coaccessible)
- pour tous q et q' distincts, $L_q \cap L_{q'} = \emptyset$ (car $\mathcal{B}$ codeterministe)

Un état de $\mathcal{B}_{det}$ correspond à un ensemble d'états $P \subset Q$.
La déterminisation garantit que le résidu associé à l'état P vérifie $L_P = \cup_{q \in P} L_q$.
D'après ce qui précède, dans $\mathcal{B}_{det}$, on a $L_P = L_{P'}$ implique $P = P'$. L'automate $\mathcal{B}_{det}$ est donc bien minimal.

Le Théorème de Brzozowski pour la minimisation

Proposition

Soit L un langage et $\mathcal{B}$ un automate co-déterministe co-accessible (ie son miroir est deterministe et accessible) qui reconnait L.
Alors le determinisé $\mathcal{B}_{det}$ de $\mathcal{B}$ est l'automate minimal de L.

Si $\mathcal{A}$ est un automate quelconque, alors $\mathcal{B} = mir(det(mir\mathcal{A}))$ est bien un automate co-déterministe et co-accessible. En appliquant la proposition précédente, on obtient bien le théorème annoncé.

Théoreme

Si L est un langage rationnel reconnu par l'automate $\mathcal{A}$, alors l'automate minimal $\mathcal{A}_L$ de L est donné par la formule suivante

$$\mathcal{A}_L = det(mir(det(mir(\mathcal{A}))))$$

Automates - Recap des Algos