Alphabet et Mots

Alphabet

- un alphabet est un ensemble ***fini*** et ***non vide***
- ses éléments sont appelés ***lettres***

Mot

- un mot défini sur un alphabet X est une ***séquence finie*** de lettres de X
- pour désigner le **mot vide** (séquence vide) on utilise le symbole ε

Longueur d'un mot

Si u est un mot et x une lettre de l'alphabet X

- $|u|$ désigne sa **longueur** (nombre de lettres)
- $|u|_x$ désigne le nombre d'occurrences de x dans u.

Alphabet et Mots

Concaténation (de mots)

- la concaténation de 2 mots u et v est notée $u.v$
- Si $u = x_1x_2...x_n$ et $v = y_1y_2...y_m$, alors $u.v = z_1z_2....z_{n+m}$ où $z_i = x_i$ pour $i \leq n$ et $z_i = y_{i-n}$ pour $i > n$
- $u.\varepsilon = \varepsilon.u = u$ (ε est **élément neutre**)

La concaténation est associative

- $(u.v).w = u.(v.w)$ On peut donc noter $u.v.w$

Propriétés

- $|u.v| = |u| + |v|$
- $\forall x \in X, |u.v|_x = |u|_x + |v|_x$

Alphabet et Mots

Itération de la concaténation

La notation u^i (où u est un mot et i un entier naturel) désigne le mot défini par

- $u^0 = \varepsilon$
- $u^{i+1} = u^i.u = u.u^i$ pour $i \geq 0$

Facteur

- un mot u est un **facteur** du mot v ssi il existe des mots p et s tels que $v = p.u.s$
- Si $\exists s$ tq $v = u.s$, alors u est un **facteur gauche** (ou **préfixe**) de v.
- Si $\exists p$ tq $v = p.u$, alors u est un **facteur droit** (ou **suffixe**) de v.
- Si u est facteur de v et $u \neq \varepsilon$ et $u \neq v$, alors u est un **facteur propre** de v.

Langages

Langage

- Un **langage** sur un alphabet X est un **ensemble de mots** sur X.
- NB : un langage n'est **pas nécessairement** un ensemble fini

Concaténation (de langages)

Si L et L' sont des langages sur X,

$$L.L' = \{u.v | u \in L, v \in L'\}$$

La concaténation est associative

$L_1.(L_2.L_3) = (L_1.L_2).L_3 = L_1.L_2.L_3$

Langages

Élément neutre

$\{\varepsilon\}$ est l'élément neutre de la concaténation de langages.

$$\forall L,\ \{\varepsilon\}.L = L.\{\varepsilon\} = L$$

Élément absorbant

$\emptyset$ est l'élément absorbant de la concaténation de langages.

$$\forall L,\ \emptyset.L = L.\emptyset = \emptyset$$

Langages

Itération de la concaténation

- $L^0 = \{\varepsilon\}$
- $L^{i+1} = L^i.L = L.L^i, \quad i \geq 0$

Clôture de Kleene

- $\bigcup_{i \geq 0} L^i$ est la **clôture de Kleene**
- On note $\mathbf{L^* = \bigcup_{i \geq 0} L^i}$ (notation « étoile de Kleene »)

- $\{\varepsilon\}^* = \{\varepsilon\}$
- $\emptyset^* = \{\varepsilon\}$, par définition.
- si L contient un mot non vide, alors L^* est infini.

Langages

Monoïde

- Si X est un alphabet, $X^* = \{mots\ sur\ l'alphabet\ X\}$
- X^* est donc une expression simple pour désigner l'ensemble des mots utilisant l'alphabet X
- X^* est appelé **monoïde libre** engendré par X.

« étoile stricte »

- On note $\mathbf{L^+} = \bigcup_{\mathbf{i \geq 1}} \mathbf{L^i}$

Autre définition, équivalente :

- $L^+ = L^*.L = L.L^*$

Langages rationnnels

Expression rationnelle

Un expression rationnelle et définie inductivement par

- atomes : $\emptyset$, ε, x (pour toute lettre $x \in X$) sont des **expressions rationnelles**
- opérateurs : $+$, $.$, *
  Si e_1 et e_2 sont des expressions rationnelles, alors
  - $e_1 + e_2$ est une expression rationnelle.
  - $e_1.e_2$ est une expression rationnelle.
  - ${e_1}^*$ est une expression rationnelle.
- parenthésage : si e est une exp. rationnelle, alors (e) est une expression rationnelle.

Langages rationnnels

Langage associé à une expression rationnelle

À chaque expression rationnelle e est associé un langage $\mathcal{L}(e)$ défini par

1. $\mathcal{L}(\emptyset) = \emptyset$, $\mathcal{L}(\varepsilon) = \{\varepsilon\}$, $\mathcal{L}(x) = \{x\}\ \forall x \in X$
2. $\mathcal{L}((e_1)) = \mathcal{L}(e_1)$
3. $\mathcal{L}(e_1 + e_2) = \mathcal{L}(e_1) \cup \mathcal{L}(e_2)$
4. $\mathcal{L}(e_1.e_2) = \mathcal{L}(e_1).\mathcal{L}(e_2)$
   si $\nexists e'$ *et* e'' *tq* $e_1 = e' + e''$ *ou* $e_2 = e' + e''$
5. $\mathcal{L}({e_1}^*) = \mathcal{L}(e_1)^*$
   si $\nexists e'$ *et* e'' *tq* $e_1 = e' + e''$ ou $e_1 = e'.e''$

$\mathcal{L}(e)$ est appelé **langage dénoté par e**

Langages rationnnels

Variantes d'expressions rationnelles :

Les expressions rationnelles sont souvent « étendues » de la façon suivante :

- Omission possible de l'opérateur point :
  $\mathcal{L}(e_1 e_2) = \mathcal{L}(e_1.e_2)$
- opérateur ^{i} (exposant) pour $i \geq 0$ (même priorité que *)
  $\mathcal{L}({e_1}^i) = \mathcal{L}(e_1)^i$
  si $\nexists e'$ *et* e'' *tq* $e_1 = e' + e''$ ou $e_1 = e'.e''$
- opérateur $^+$ (comme exposant, ne pas confondre avec $+$)
  $\mathcal{L}({e_1}^+) = \mathcal{L}(e_1)^+ = \mathcal{L}(e_1).\mathcal{L}(e_1)^*$
  si $\nexists e'$ *et* e'' *tq* $e_1 = e' + e''$ ou $e_1 = e'.e''$

NB : Les expressions implémentées dans les composants logiciels ont leur propre syntaxe (cf TDM)

Langages rationnels

Définition 1

- Un **langage rationnel** est un langage qui peut être dénoté par une expression rationnelle
- On appelle RAT la **famille des langages rationnels**

Définition 2 ou propriété

- RAT est la plus petite famille de langages qui
  - contient $\emptyset, \{\varepsilon\}, \{x\}, \forall x \in X$
  - est close par union, concaténation et étoile de Kleene

Propriété

- Tout langage fini $\in RAT$

Propriété

- Il existe des langages non rationnels (la preuve sera apportée plus tard).

Automates finis déterministes

Définition

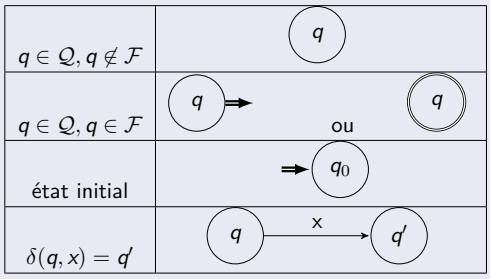
Un automate fini déterministe est défini par

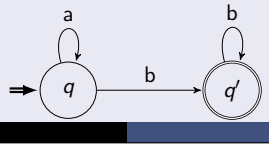
- un alphabet X
- un ensemble fini $\mathcal{Q}$ appelé **ensemble d'états**
- un **état initial** $q_{ini} \in \mathcal{Q}$
- un sous-ensemble $\mathcal{F} \subseteq \mathcal{Q}$ d'états dits **acceptants** (ou encore **finals** ou encore **terminaux**)
- une fonction $\delta : \mathcal{Q} \times X \to \mathcal{Q}$ appelée **fonction de transition**

- Notation : $\delta(q_d, x) = q_a$ pourra être noté $q_d \xrightarrow{x} q_a$
- q_d est appelé état de départ et q_a état d'arrivée

Automates finis déterministes

Représentation graphique





Automates finis déterministes

Extension de la fonction de transition

La fonction de transition peut être étendue aux mots :

$$\begin{array}{rcl} \hat{\delta} : \mathcal{Q} \times X^* & \rightarrow & \mathcal{Q} \\ (q, \varepsilon) & \mapsto & q \\ (q, xw) & \mapsto & \hat{\delta}(\delta(q, x), w)\ x \in X, w \in X^* \end{array}$$

Autre définition équivalente

$$\begin{array}{rcl} \hat{\delta} : \mathcal{Q} \times X^* & \rightarrow & \mathcal{Q} \\ (q, \varepsilon) & \mapsto & q \\ (q, wx) & \mapsto & \delta(\hat{\delta}(q, w), x)\ x \in X, w \in X^* \end{array}$$

- si $q' = \hat{\delta}(q, w)$ on pourra utiliser la notation $q \xrightarrow[*]{w} q'$
- soit $u = x_1 x_2 ... x_{n+1}$, $x_i \in X$, et $r_i \in \mathcal{Q}$ tq $r_i \xrightarrow{x_i} r_{i+1}$, alors $r_1 \xrightarrow[*]{u} r_{n+1}$

Automates finis déterministes

Langage reconnu

Le langage reconnu par un automate déterministe $\mathcal{A} = (X, \mathcal{Q}, \delta, q_{ini}, \mathcal{F})$ est :

$$\mathcal{L}(\mathcal{A}) = \{w \in X^* | \hat{\delta}(q_{ini}, w) \in \mathcal{F}\}$$

Remarque : propriété

$\varepsilon \in \mathcal{L}(\mathcal{A}) \iff q_{ini} \in \mathcal{F}$

Automates finis déterministes

Automate déterministe complet

Un automate déterministe A est dit **complet** si sa fonction de transition δ est définie $\forall (q, x) \in \mathcal{Q} \times X$

Propriété

Tout langage reconnu par un automate peut être reconnu par un automate complet.

Si un automate déterministe A n'est pas complet, il existe un algorithme simple permettant de construire un automate déterministe complet qui reconnaît le même langage que A

Automates finis déterministes : algo de reconnaissance

Require : A est un automate fini déterministe
Require : u est un mot quelconque.
Si $u \neq \varepsilon$, on appelle x_i $(0 \leq i < |u|)$ sa lettre de rang i
Ensure : Renvoie TRUE si et seulement si u est accepté par A.

$etatCourant \leftarrow$ état initial de A ;
$index = 0$;
while $(index < |u|)$ **do**
  $lettre \leftarrow x_{index}$;
  if ($\delta(etatCourant, lettre)$ est indéfini) **then**
    return FALSE ;
  end if
  $etatCourant \leftarrow \delta(etatCourant, lettre)$;
  $index \leftarrow index + 1$;
end while
return $etatCourant \overset{?}{\in} \mathcal{F}$;

Automates finis déterministes

État accessible

Soit A, un automate. Un état q est dit **accessible** s'il existe un mot w tel que $\hat{\delta}(q_{ini}, w) = q$

Automate accessible

Un automate est dit **accessible** si tous ses états sont accessibles.

Propriété

Tout langage reconnu par un automate peut être reconnu par un automate accessible

État co-accessible

Soit A, un automate. Un état q est dit **co-accessible** s'il existe un mot w tel que $\hat{\delta}(q, w) \in \mathcal{F}$

Automates finis déterministes

Automate co-accessible

Un automate est dit **co-accessible** si tous ses états sont co-accessibles.

Propriété

Tout langage reconnu par un automate peut être reconnu par un automate co-accessible

Automate émondé

Un automate est dit **émondé** s'il est **accessible ET co-accessible**

Propriété

Tout langage reconnu par un automate peut être reconnu par un automate émondé

Automates finis non déterministes

Définition

Un automate fini non déterministe est défini par

- un alphabet X
- un ensemble fini $\mathcal{Q}$ appelé **ensemble d'états**
- un **ensemble non vide d'états initiaux** : $Ini \in 2^{\mathcal{Q}}, Ini \neq \emptyset$
- un sous-ensemble $\mathcal{F} \subseteq \mathcal{Q}$ d'états dits **acceptants** (ou encore **finals** ou encore **terminaux**)
- une fonction $\delta : \mathcal{Q} \times X \to 2^{\mathcal{Q}}$ appelée **fonction de transition**

- Notation : $q_a \in \delta(x, q_d)$ pourra être noté $q_d \xrightarrow{x} q_a$

Automates finis non déterministes

Extension de la fonction de transition

La fonction de transition peut être étendue aux mots :

$$\begin{array}{rcl} \hat{\delta} : \mathcal{Q} \times X^* & \to & 2^{\mathcal{Q}} \\ (q, \varepsilon) & \mapsto & \{q\} \\ (q, xw) & \mapsto & \bigcup_{q' \in \delta(q,x)} \hat{\delta}(q', w) \ (x \in X, w \in X^*) \end{array}$$

Langage reconnu

Le langage reconnu par un automate non déterministe $\mathcal{A} = (X, \mathcal{Q}, \delta, Ini, \mathcal{F})$ est défini par

$$\mathcal{L}(\mathcal{A}) = \{w \in X^* | \exists q_{ini} \in Ini, \hat{\delta}(q_{ini}, w) \cap \mathcal{F} \neq \emptyset\}$$

Équivalence entre automates finis déterministes et non déterministe

$REC \subseteq REC_{ND}$

Pour tout automate fini déterministe A, il existe un automate fini non déterministe A_{nd} qui reconnaît le même langage

A_{nd} peut être défini de façon triviale par

- $\mathcal{Q}_{A_{nd}} = \mathcal{Q}_A$
- $Ini_{A_{nd}} = \{q_{ini}\}$
- $\mathcal{F}_{A_{nd}} = \mathcal{F}_A$
- $\delta_{A_{nd}}(q, x) = \{\delta_A(q, x)\}$ si $\delta_A(q, x)$ est défini.
- $\delta_{A_{nd}}(q, x) = \emptyset$ si $\delta_A(q, x)$ est indéfini.

Équivalence entre automates finis déterministes et non déterministe

$REC_{ND} \subseteq REC$

Pour tout automate fini **non** déterministe A, il existe un automate fini déterministe A_d qui reconnaît le même langage

Automate déterministe équivalent

Pour un automate non déterministe A, un automate déterministe équivalent A_d peut être défini par

- $\mathcal{Q}_{A_d} = 2^{\mathcal{Q}_A}$
- $q_{ini} = Ini_A$
- $\mathcal{F}_{A_d} = \{q \in \mathcal{Q}_{A_d}, q \cap \mathcal{F}_A \neq \emptyset\}$
- $\delta_{A_d}(q, x) = \bigcup_{e \in q} \delta_A(e, x)$

Si L est reconnaissable, on pourra donc toujours supposer qu'il existe un automate à **un seul** état initial qui le reconnaît.

Déterminisation

Algorithme

$q_{ini} \leftarrow Ini_A$
$\mathcal{Q}_{A_d} \leftarrow \{Ini_A\}$
while $\exists (q, x) \in \mathcal{Q}_{A_d} \times X$, $\delta_{A_d}(q, x)$ *is undefined* **do**
  $(q, x) \leftarrow$ one of such pair
  $q' \leftarrow \bigcup_{e \in q} \delta_A(e, x)$
  $\mathcal{Q}_{A_d} \leftarrow \mathcal{Q}_{A_d} \cup \{q'\}$
  $\delta_{A_d}(q, x) \leftarrow q'$
end while
$\mathcal{F}_{A_d} \leftarrow \{q \in \mathcal{Q}_{A_d}, q \cap \mathcal{F}_A \neq \emptyset\}$

- L'automate obtenu est **complet** et **complètement accessible**.
- Il n'est, en général, **pas minimal**

Stabilité de REC par $\cup, ., ^*$

Théorème

REC est close par union, concaténation et étoile de Kleene.

Clôture par union

Soient $L_1 \in REC$, reconnu par un automate A_1 et $L_2 \in REC$, reconnu par un automate A_2. On suppose que $\mathcal{Q}_{A_1} \cap \mathcal{Q}_{A_2} = \emptyset$ (il suffit de renommer les états si nécessaire)
L'automate non déterministe défini par

- $\mathcal{Q}_A = \mathcal{Q}_{A_1} \cup \mathcal{Q}_{A_2}$
- $Ini_A = Ini_{A_1} \cup Ini_{A_2}$
- $\mathcal{F}_A = \mathcal{F}_{A_1} \cup \mathcal{F}_{A_2}$
- $\forall q \in \mathcal{Q}_{A_1},\ \delta_A(q,x) = \delta_{A_1}(q,x)$
- $\forall q \in \mathcal{Q}_{A_2},\ \delta_A(q,x) = \delta_{A_2}(q,x)$

reconnait le langage $L_1 \cup L_2$
Donc $L_1 \cup L_2 \in REC$.

Stabilité de REC par $\cup, ., ^*$

Clôture par concaténation

Soient $L_1 \in REC$, reconnu par un automate A_1 et $L_2 \in REC$, reconnu par un automate A_2. On suppose que $\mathcal{Q}_{A_1} \cap \mathcal{Q}_{A_2} = \emptyset$
On suppose, de plus, que A_2 possède un seul état initial : q_{iniA2}
L'automate non déterministe défini par

- $\mathcal{Q}_A = \mathcal{Q}_{A_1} \cup \mathcal{Q}_{A_2}$
- $Ini_A = Ini_{A_1}$
- $\mathcal{F}_A = \mathcal{F}_{A_2} \cup_{Seulement\ si\ \varepsilon \in L_2} \mathcal{F}_{A_1}$
- $\forall q \in \mathcal{Q}_{A_1} \backslash \mathcal{F}_{A_1},\ \delta_A(q,x) = \delta_{A_1}(q,x)$
- $\forall q \in \mathcal{F}_{A_1},\ \delta_A(q,x) = \delta_{A_1}(q,x) \cup \delta_{A_2}(q_{iniA2},x)$
- $\forall q \in \mathcal{Q}_{A_2},\ \delta_A(q,x) = \delta_{A_2}(q,x)$

reconnait le langage $L_1.L_2$
Donc $L_1.L_2 \in REC$.

Stabilité de REC par $\cup, ., ^*$

Clôture par étoile de Kleene

Soient $L_1 \in REC$, tel que $\varepsilon \in L$, reconnu par un automate A_1 possédant un seul état initial q_{iniA1} L'automate non déterministe défini par

- $\mathcal{Q}_A = \mathcal{Q}_{A_1}$
- $Ini_A = Ini_{A_1} = \{q_{iniA1}\}$
- $\mathcal{F}_A = \mathcal{F}_{A_1}$
- $\forall q \in \mathcal{Q}_{A_1} \backslash \mathcal{F}_{A_1},\ \delta_A(q,x) = \delta_{A_1}(q,x)$
- $\forall q \in \mathcal{F}_{A_1},\ \delta_A(q,x) = \delta_{A_1}(q,x) \cup \delta_{A_1}(q_{iniA1},x)$

reconnait le langage L_1^*
Donc $L_1^* \in REC$.

Si $L \in REC$ et $\varepsilon \notin L$, on remarque que $L^* = (L \cup \{\varepsilon\})^*$ et que $(L \cup \{\varepsilon\}) \in REC$. Donc $L^* \in REC$ d'après ce qui précède.

Stabilité de REC par $\cup, ., ^*$

Théorème

$RAT \subseteq REC$

- *REC* est close par union, concaténation et étoile de Kleene.
- *REC* contient $\emptyset, \{\varepsilon\}, \{x\}\ (\forall x \in X)$
- *RAT* est la plus petite famille close par union, concaténation et étoile de Kleene contenant $\emptyset, \{\varepsilon\}, \{x\}\ (\forall x \in X)$
- $RAT \subseteq REC$

Tout langage rationnel peut être reconnu par un automate.

Équations et langages

Soient A et B deux langages. L'équation à une inconnue (notée L)

$$L = A.L \cup B$$

- Admet-elle une solution ?
- Si oui, est-elle unique ?
- Comment la calculer ?

Équations et langages

Lemme d'Arden

Soient A et B deux langages, et l'équation

$$L = A.L \cup B$$

- $A^*.B$ est une solution.
- $A^*.B$ est une solution minimale.
- si $\varepsilon \notin A$, alors $A^*.B$ est l'**unique** solution .

Remarque : Si $A \in RAT$, $B \in RAT$ et $\varepsilon \notin A$, l'unique solution $A^*B \in RAT$

Équations et langages

Système d'équations

Un système à n inconnues L_i, $(1 \leq i \leq n)$ et n équations

$$(i) \qquad L_i = \bigcup_{j=1}^{n} A_{i,j}.L_j \ \cup \ B_i$$

où les $A_{i,j}$ et les B_i sont des langages et où $\varepsilon \notin A_{i,j}$

- admet une solution unique.
- si les $A_{i,j}$ et B_i sont rationnels, alors chaque L_i est rationnel.

Équations et langages

Notation

Si les $A_{i,j}$ et B_i sont rationnels, alors on pourra

- utiliser le signe $+$ à la place de $\cup$
- utiliser des expressions rationnelles à la place des langages

En d'autres termes, la notation :

$$(i) \quad L_i = e_{i,1}L_1 + e_{i,2}L_2 + \ldots e_{i,n}L_n + e'_i$$

où les $e_{i,j}$ et e'_i sont des expressions rationnelles a la même signification que

$$(i) \quad L_i = \mathcal{L}(e_{i,1}).L_1 \cup \mathcal{L}(e_{i,2}).L_2 \cup \ldots \mathcal{L}(e_{i,n}.)L_n \cup \mathcal{L}(e'_i)$$

Équations et langages

Langages L_q, définition

Soit un automate $\mathcal{A} = (X, \mathcal{Q}, q_{ini}, \mathcal{F}, \delta)$ et un état $q \in Q$,

$$L_q = \{w \in X^* | \hat{\delta}(q, w) \in \mathcal{F}\}$$

Propriété

$$L_{q_{ini}} = \mathcal{L}(\mathcal{A})$$

Système d'équations des L_q (cas de l'automate déterministe)

- si $q \notin \mathcal{F}$:
$$L_q = \bigcup_{x \in X} \{x\}.L_{\delta(q,x)}$$
- si $q \in \mathcal{F}$:
$$L_q = \{\varepsilon\} \cup \bigcup_{x \in X} \{x\}.L_{\delta(q,x)}$$

Équations et langages

Système d'équations associées à un automate déterministe (définition)

À tout automate déterministe A, on associe un système de $card(\mathcal{Q})$ équations à $card(\mathcal{Q})$ inconnues (notées ici L_q).
(e_q) :

- $L_q = \bigcup_{x \in X} \{x\}.L_{\delta(q,x)}$, si $q \notin \mathcal{F}$
- $L_q = \{\varepsilon\} \cup \bigcup_{x \in X} \{x\}.L_{\delta(q,x)}$, si $q \in \mathcal{F}$:

Proposition

- Le système d'équations admet une solution unique.
- La solution pour chaque L_q est un langage rationnel.

Méthode de résolution

Système d'équations linéaire et utilisation du lemme d'Arden

Théorême de Kleene : $REC = RAT$

$REC \subseteq RAT$

Pour tout automate déterministe A

- Chaque L_q admet une solution unique, qui est un langage rationnel.
- $\mathcal{L}(A) = L_{q_{ini}}$ est un langage rationnel

Donc $REC \subseteq RAT$

Théorème de Kleene

$$RAT = REC$$

Résiduels

Langage résiduel : définition

Pour tout langage $L \subseteq X^*$ et tout mot $u \in X^*$ on appelle **langage résiduel de L par u** le langage

$$u^{-1}L = \{v \in X^* | \ u.v \in L\}$$

Propriétés « constructives » des résiduels

$\forall u \in X^*, \forall x \in X, \forall L \subseteq X^*$,

$$(u.x)^{-1}L = x^{-1}(u^{-1}L) \quad (1)$$
$$u^{-1}(L_1 \cup L_2) = u^{-1}L_1 \cup u^{-1}L_2 \quad (2)$$
$$x^{-1}(L_1.L_2) = (x^{-1}L_1).L_2 \ \cup \ Eps(L_1).(x^{-1}L_2) \quad (3)$$
$$x^{-1}L^* = (x^{-1}L).L^* \quad (4)$$

où $Eps(L) = L \cap \{\varepsilon\}$

Équations et langages

Propriété

Soit L un langage et $u \in L, u \neq \varepsilon$. u s'écrit $x.v$ où x est sa 1ère lettre.
$u = x.v \in L \iff v \in x^{-1}L \iff u \in x.(x^{-1}L)$
En conséquence,

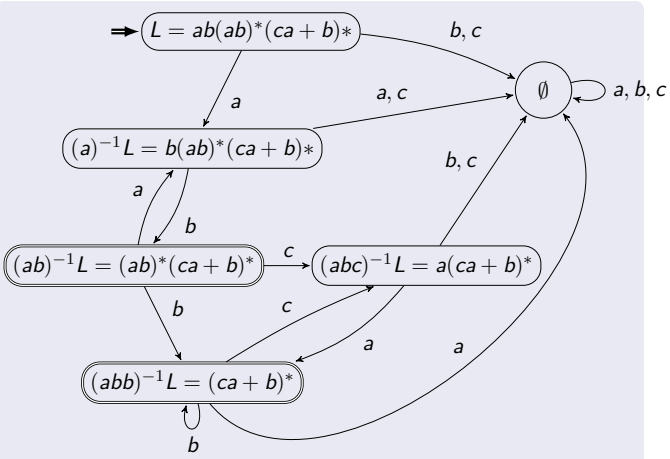
- si $\varepsilon \notin L$:
$$L = \bigcup_{x \in X} x.(x^{-1}L)$$
- si $\varepsilon \in L$:
$$L = \bigcup_{x \in X} x.(x^{-1}L) \cup \{\varepsilon\}$$

.

Exemple de calcul des résiduels

$$\begin{aligned}
L &= ab(ab)^*(ca+b)* \\
a^{-1}L &= b(ab)^*(ca+b)* \\
(aa)^{-1}L &= \emptyset \\
(ab)^{-1}L &= (ab)^*(ca+b)^* \\
(ac)^{-1}L &= \emptyset \\
(aba)^{-1}L &= b(ab)^*(ca+b)^* && = (a)^{-1}L \\
(abb)^{-1}L &= (ca+b)^* \\
(abc)^{-1}L &= a(ca+b)^* \\
(abba)^{-1}L &= \emptyset \\
(abbb)^{-1}L &= (ca+b)^* && = (abb)^{-1}L \\
(abbc)^{-1}L &= a(ca+b)^* && = (abc)^{-1}L \\
(abca)^{-1}L &= (ca+b)^* && = (abb)^{-1}L \\
(abcb)^{-1}L &= \emptyset
\end{aligned}$$

Automate des résiduels



Automate des résiduels

Automate des résiduels

Soit $L \subseteq X^*$ *possédant un nombre fini de résiduels*, on définit son **automate des résiduels** (automate déterministe)

- $\mathcal{Q} = \{u^{-1}L, u \in X^*\}$ (ensemble (fini) des résiduels de L)
- $q_{ini} = L = \varepsilon^{-1}L$
- $\mathcal{F} = \{u^{-1}L \in \mathcal{Q},\ \varepsilon \in u^{-1}L\} = \{u^{-1}L,\ u \in L\}$
- $\forall (u^{-1}L) \in \mathcal{Q},\ \forall x \in X,\ \delta(u^{-1}L, x) = (ux)^{-1}L = x^{-1}(u^{-1}L)$

Propriété

Si L possède un nombre fini de résiduels, l'automate des résiduels de L reconnaît L.

Propriété

Tout langage possédant un nombre fini de résiduels est reconnaissable.

Résiduels et automates

Propriété

Soit A, automate **déterministe**, $q \in Q$ un état et $u \in X^*$ un mot :

$$\hat{\delta}(q_{ini}, u) = q \Rightarrow L_q = u^{-1}\mathcal{L}(A)$$

Propriété

Soit automate **déterministe et accessible** reconnaissant L, chaque langage L_q est un résiduel de L.

Propriété

Dans un automate **déterministe complet** reconnaissant L, pour tout résiduel R de L, il existe au moins un état q tel que $L_q = R$.

Propriété

Tout automate **déterministe complet** possède une nombre d'états **au moins égal au nombre de résiduels du langage reconnu**.

Résiduels et reconnaissables

Propriété

Tout langage reconnaissable admet un nombre fini de résiduels

Propriété

Un langage est reconnaissable si et seulement si il admet un nombre fini de résiduels

Propriété

Tout langage reconnaissable admet un automate déterministe complet minimal unique.
Cet automate est son automate des résiduels.

Minimalisation

Congruence de Nérode

Définie sur les états d'un automate déterministe et accessible par

$$p \cong q \iff L_p = L_q$$

C'est bien une congruence

- compatible avec δ : $\forall x \in X, p \cong q \iff \delta(p,x) \cong \delta(q,x)$
- sature $\mathcal{F}$: $p \cong q \Rightarrow (p \in \mathcal{F} \iff q \in \mathcal{F})$

Nombre de classes d'équivalence

Autant de classes d'équivalence que d'ensembles L_q distincts, donc autant que de langages résiduels

Conclusion

L'automate quotient est l'automate minimal

Minimalisation, méthode de Moore

Algorithme de Moore : calcul de la congruence de nérode

Calcul récurrent

1. $$p \cong_0 q \iff (p \in \mathcal{F} \iff q \in \mathcal{F})$$

   2 états sont équivalents (niveau 0) s'ils sont tous deux acceptants ou tous deux non acceptants.

2. $$p \cong_{i+1} q \iff p \cong_i q \text{ et } \forall x \in X, \delta(p,x) \cong_i \delta(q,x)$$

   2 états sont équivalents au niveau $i+1$ s'ils le sont au niveau i **ET** si pour toute lettre x les états obtenus par x depuis ces 2 états sont équivalents au niveau i.

- le calcul est itéré jusqu'à n tel que $\cong_n = \cong_{n-1}$
- $\cong \;=\; \cong_n$

Minimalisation, méthode de Moore

Partition de l'ensemble d'états par raffinements successifs

Les états sont répartis en classes qui sont raffinées à chaque étape.
On note $[q]_i$ la classe de q à l'étape i.
On note $x_1, x_2, ...x_m$ les lettres de l'alphabet.

- au départ ($i = 0$), 2 classes : $\{\mathcal{F}, Q - \mathcal{F}\}$
- passer de l'étape i à $i+1$:
  Pour **chaque classe**, établir le « profil » de chaque état :

  $$profil_i(q) = ([\delta(q,x_1)]_i\ ,\ [\delta(q,x_2)]_i\ ,\ \ldots\ ,\ [\delta(q,x_m)]_i)$$

  si plusieurs profils distincts apparaissent, la classe est scindée : une partie par profil, regroupant les états de même profil.
- fin quand aucune partie n'a été scindée lors d'une étape.
- les classes obtenues sont les classes de l'équivalence de Nérode
- l'automate minimal est l'automate « quotient »

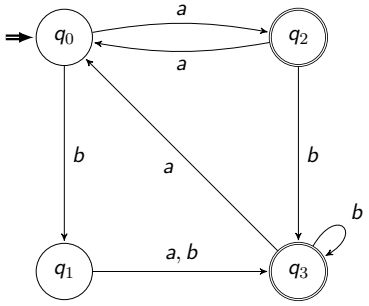
Minimalisation, méthode de Moore

Automate minimal

- $\mathcal{Q}^{min} \;=\; \{\ [q]\ , q \in Q\}$
- $q_{ini}{}^{min} \;=\; [q_{ini}]$
- $\mathcal{F}^{min} \;=\; \{\ [q]\ , [q] \subseteq \mathcal{F}\}$
- $\delta([q], x) \;=\; [\ \delta(q,x)\ ]$

Exemple de minimalisation (algo de Moore)

Automate initial
(déterministe complet et accessible)



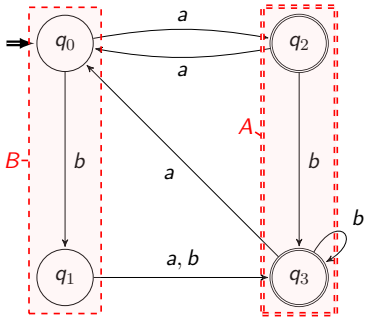
Étape 0

Créer 2 classes A et B

- une pour les états acceptants
- une pour tous les autres

Exemple de minimalisation (algo de Moore)

Étape 0 (résultat)
$A = \{q_2, q_3\} \subseteq \mathcal{F},\ B = \{q_0, q_1\}$



Étape 1

A	a	b
q_2	B	A
q_3	B	A

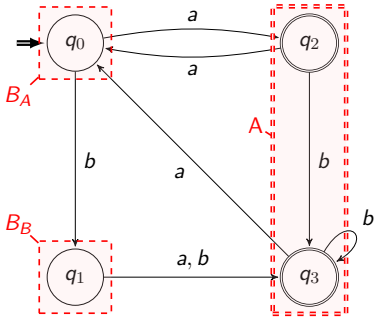
q_2 et q_3 ont même profil.
A n'est pas scindée.

B	a	b
q_0	A	B
q_1	A	A

q_0 et q_1 ont des profils distincts (donc sont **non** équivalents).
On scinde B en 2 classes :
$B_A = \{q_0\},\ B_B = \{q_1\}$

Exemple de minimalisation (algo de Moore)

Étape 1 (résultat)
$A = \{q_2, q_3\} \subseteq \mathcal{F}$,
$B_A = \{q_0\},\ B_B = \{q_1\}$



Étape 2

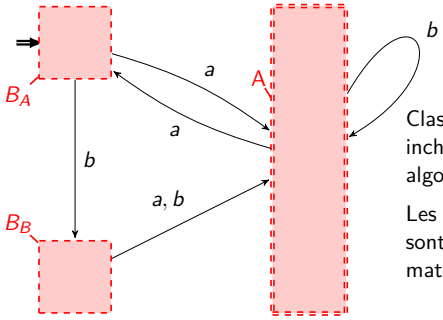
A	a	b
q_2	B_A	A
q_3	B_A	A

q_2 et q_3 sont équivalents.
A n'est pas scindée.

Les classes B_A et B_B ne peuvent pas être scindées (singletons).

Exemple de minimalisation (algo de Moore)

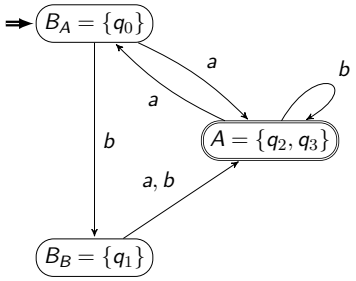
Étape 2 (résultat)
$A = \{q_2, q_3\} \subseteq \mathcal{F}$,
$B_A = \{q_0\},\ B_B = \{q_1\}$



Classes d'équivalence inchangées, donc algorithme terminé.

Les classes obtenues sont les états de l'automate minimal.

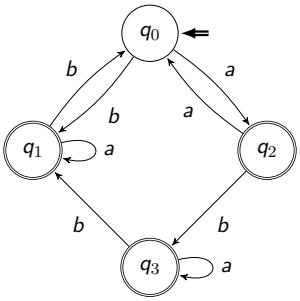
Exemple de minimalisation (algo de Moore)



Automate déterministe complet minimal

Exemple de minimalisation (algo de Moore)

Automate initial
(déterministe complet et accessible)



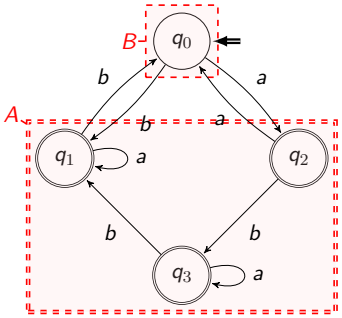
Étape 0

Créer 2 classes A et B

- une pour les états acceptants
- une pour tous les autres

Exemple de minimalisation (algo de Moore)

Étape 0 (résultat)
$A = \{q_1, q_2, q_3\} \subseteq \mathcal{F},\ B = \{q_0\}$



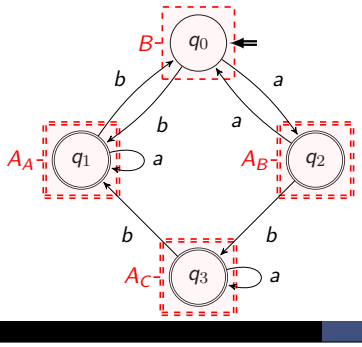
Étape 1

A	a	b
q_1	A	B
q_2	B	A
q_3	A	A

Les états présentent 3 profils distincts :
$(A, B),\ (B, A),\ (A, A)$
A est scindée en 3.

Exemple de minimalisation (algo de Moore)

Étape 1 (résultat)
$A_A = \{q_1\},\ A_B = \{q_2\},\ A_C = \{q_3\}$
$A_A \subseteq \mathcal{F},\ A_B \subseteq \mathcal{F},\ A_C \subseteq \mathcal{F}$
$B = \{q_0\}$



Étape 2

Plus aucun raffinement n'est possible..

NB : on est revenu à l'automate initial, qui était donc minimal