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 ***suite finie*** de lettres de X
- pour désigner le **mot vide** (suite 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 $u = x_1 x_2 ... x_n$ et $v = y_1 y_2 ... y_m$, alors
  $u.v = z_1 z_2 .... z_{n+m}$
  où $z_i = x_i$ pour $i \leq n$ et $z_i = y_{i-n}$ pour $i > n$
- $u.\epsilon = \epsilon.u = u$ (ϵ est **élément neutre**)

La concaténation est associative

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

propriété

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

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

- $u^0 = \epsilon$
- $u^{i+1} = u^i.u$ 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 $u \neq \epsilon$ 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

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

Élément neutre

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

Itération de la concaténation

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

Clôture de Kleene

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

« étoile stricte »

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

Monoïde

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

Expression rationnelle

Un expression rationnelle et définie inductivement par

- $\emptyset$ $\quad\epsilon\quad$ et x sont des expressions rationnelles ($x \in X$)
- Si e_1 et e_2 sont des expressions rationnelles, alors
  - (e_1) est une expression rationnelle.
  - $e_1 + e_2$ est une expression rationnelle.
  - $e_1.e_2$ est une expression rationnelle.
  - e_1^* est une expression rationnelle.

NB : on peut aussi utiliser aussi e^+ comme équivalent de $e^*.e$ dans les expressions rationnelles (mais attention à la confusion avec l'autre opérateur $+$)

Langage associé à une expression rationnelle

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

1. $\mathcal{L}(\emptyset) = \emptyset$, $\mathcal{L}(\epsilon) = \{\epsilon\}$, $\mathcal{L}(x) = \{x\}\ \forall x \in X$
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* $e_2 = e' + e''$
   ou $e_1 = e'.e''$ *ou* $e_2 = e'.e''$

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

Langages rationnnels

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 $\emptyset, \{\epsilon\}, \{x\}, \forall x \in X$
  - est close par union, concaténation et étoile de Kleene :

Automates finis déterministes

Définition

Un automate fini déterministe est défini par

- un alphabet X
- un ensemble fini $\mathcal{Q}$ appelé **ensemble d'états**
- une fonction $\delta : \mathcal{Q} \times X \to \mathcal{Q}$ appelée **fonction de transition**
- 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**)

- Notation : $\delta(x, q_d) = q_a$ 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

Extension de la fonction de transition

La fonction de transition peut être étendue aux mots :

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

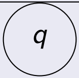
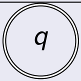
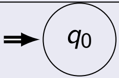
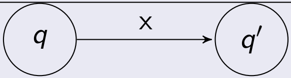
- si $q' = \hat{\delta}(q, w)$ on pourra utiliser la notation $q \xrightarrow[*]{w} q'$
- si $u = x_1 x_2 ... x_{n+1}, x_i \in X$, et $r_i \in \mathcal{Q}$ tq $r_i \xrightarrow{x_i} q_{i+1}$, alors $r_1 \xrightarrow[*]{u} r_{n+1}$

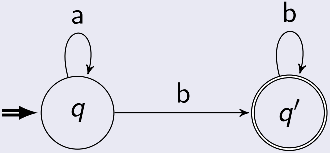
Langage reconnu

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

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

Représentation graphique

$q \in \mathcal{Q}, q \notin \mathcal{F}$	
$q \in \mathcal{Q}, q \in \mathcal{F}$	
état initial	
$\delta(q, x) = q'$	



État accessible

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

Automate complètement accessible

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

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

Automate complet

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

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

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**
- une fonction $\delta : \mathcal{Q} \times X \to 2^{\mathcal{Q}}$ appelée **fonction de transition**
- un **ensemble 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**)

- 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 :

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

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' qui reconnaît le même langage

A' peut être défini de façon triviale par

- $\mathcal{Q}_{A'} = \mathcal{Q}_A$
- $Ini_{A'} = \{q_{ini}\}$
- $\mathcal{F}_{A'} = \mathcal{F}_A$
- $\delta_{A'}(q, x) = \{\delta_A(q, x)\}$

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

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

Note : l'algorithme de déterminisation permettra d'éliminer les (a priori nombreux) états inaccessibles.

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 $L_1 \cup Ł_2$
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 $\mathcal{Q}_{A_1} \cap \mathcal{Q}_{A_2} = \emptyset$
L'automate non déterministe défini par

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

reconnait le langage $L_1.Ł_2$
Donc $L_1.L_2 \in REC$.

Stabilité de REC par $\cup, ., ^*$

Clôture par étoile de Kleene

Soient $L_1 \in REC$, tel que $\epsilon \in L$, reconnu par un automate A_1
L'automate non déterministe défini par

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

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

Si $L \in REC$ et $\epsilon \notin L$, on remarque que $L^* = (L \cup \{\epsilon\})^*$ et que $(L \cup \{\epsilon\}) \in REC$. 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 $\emptyset, \{\epsilon\}, \{x\}$ $(\forall x \in X)$
- RAT est la plus petite famille close par union, concaténation et étoile de Kleene contenant $\emptyset, \{\epsilon\}, \{x\}$ $(\forall x \in X)$
- $RAT \subseteq REC$

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 $\epsilon \notin A$, alors $A^*.B$ est l'**unique** solution .

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

Équations et langages

Rappel : pour un automate déterministe A, on pose
$L_q = \{w \in X^* | \hat{\delta}(q, w) \in \mathcal{F}\}$

Pour tout état q, L_q peut s'exprimer en fonction des autres langages $L_{q'}$:

- si $q \notin \mathcal{F}$:

$$L_q = \bigcup_{x \in X} \{x\}.L_{\delta(q,x)}$$

- si $q \in \mathcal{F}$:

$$L_q = \{\epsilon\} \cup \bigcup_{x \in X} \{x\}.L_{\delta(q,x)}$$

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

Proposition

- Le système d'équation 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

Algorithme

Require: Les états sont des entiers de 0 à $n-1$

- **for** $i = 0$ **to** $n-1$ **do**
  - **for** $j = 0$ **to** $i-1$ **do**
    - Dans (e_i), remplacer L_j par son expression (e'_j)
  - **end for**$\{ -\alpha- \}$
  - **if** L_i apparaît dans la partie droite de (e_i) **then**
    - Utiliser Arden pour éliminer L_i
  - **end if**$\{ -\beta- \}$
  - Nommer (e'_i) la nouvelle equation obtenue
- **end for**$\{(e'_{n-1})$ est une solution rationnelle pour $L_{n-1}\}$
- **for** $i = n-2$ **to** 0 **do**
  - Calculer la solution pour L_i
- **end for**$\{\forall i, (e'_i)$ est une solution pour $L_i\}$

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

$$RAT = REC$$

Rappel : 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

$$L/u = \{v \in X^* | uv \in L\}$$

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

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

Résiduels et automates

Soit $L \subseteq X^*$ un langage reconnu par un automate déterministe $\mathcal{A} = (X, \mathcal{Q}, q_{ini}, \mathcal{F}, \delta)$, alors pour tout mot u, $L/u = L_{\hat{\delta}(q_0,u)}$

Tout langage reconnaissable admet un nombre fini de résiduels

Tout automate **déterministe complet** reconnaissant un langage L possède une nombre d'états au moins égal au nombre de résiduels de L

Automate des résiduels

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

- $\mathcal{Q} = \{L/w, w \in X^*\}$ (ensemble fini, par hypothèse)
- $Q_{ini} = L = L/\epsilon$
- $\mathcal{F} = \{L/w, w \in L\}$
- $\delta(L/w, x) = L/wx$

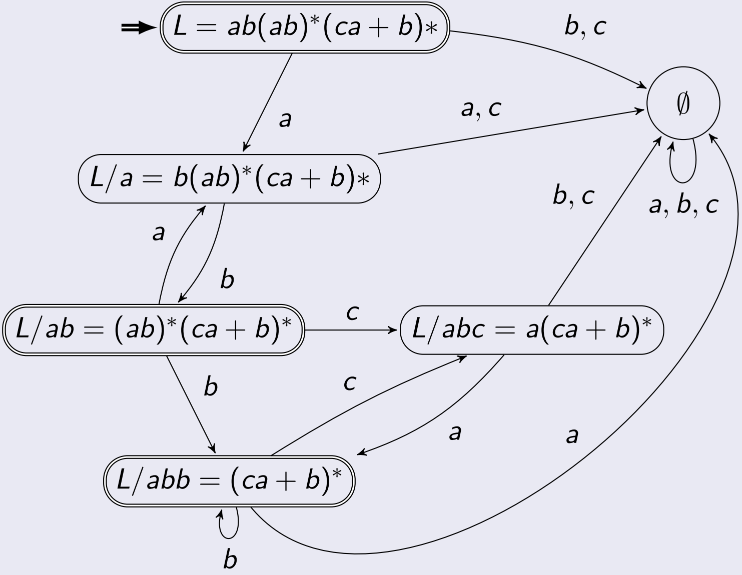
L'automate des résiduels de L reconnaît L. $\Rightarrow$ **Tout langage possédant un nombre fini de résiduels est reconnaissable**.

Si $L \in REC$, l'automate des résiduels de L est l'**automate déterministe complet minimal** parmi les automates reconnaissant L.
NB : à un renommage des états près .

Exemple de calcul des résiduels

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

Automate des résiduels



Minimalisation

Congruence de Nérode

Définie sur les états d'un automate déterministe par

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

C'est bien une congruence

- compatible avec δ : $\forall a \in X, p \cong q \iff \delta(p,x) \cong \delta(q,x)$
- 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

Algorithme de Moore : calcul de la congruence de nérode

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

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

- Le calcul est itéré jusqu'à n tel que $\cong_{n+1} = \cong_n$
- $\cong \quad = \quad \cong_n$

Partition de l'ensemble d'états par raffinements successifs

- $P_0 = \{\mathcal{F}, Q \backslash \mathcal{F}\}$
- soit P_i la partition. Appelons $x_1, ...x_n$ les lettres de X.
  Toute partie $C \in P_i$ est partitionnée en sous-parties
  $C_{E_1,E_2,...E_k} = \{q \in C, E_j \in P_i \text{ } et \text{ } \delta(q, x_j) \in E_j\}$
  La partition P_{i+1} est constituée de toutes les sous-parties
  $C_{E_1,E_2,...E_k}$ pour $C \in P_i$