Théorie des langages

Joëlle Thollot

Transcrit par Julien Henry

Ce cours n'est pas un polycopié officiel. Aussi, il n'est pas certifié sans erreurs. Pour toute remarque ou correction, vous pouvez me contacter à l'adresse

julien.henry@ensimag.imag.fr

2008-2009

École Nationale Supérieure d'Informatique et de Mathématiques Appliquées, Grenoble

Table des matières

Définitions préliminaires

Langage :	ensemble d'objets élémentaires que l'on peut combiner pour obtenir des mots qui ont un sens.	
Vocabulaire :	ensemble d'objets élémentaires	$\Rightarrow$ *Lexicographie*
Mot :	suite d'objets élémentaires	$\Rightarrow$ *Syntaxe*

Ces mots ont un certain sens, c'est ce qu'on appelle la *Sémantique*.

Chapitre I

Outils de base

1. Définitions préliminaires

Vocabulaire : Ensemble fini de symboles (lettres, caractères). On le note V.

Exemple a,b,c,0,1,begin,end,... *

Mots : ou aussi chaînes, phrases. Un mot sur un vocabulaire V est une suite finie de lettres de V.

Exemple ababc, 1001. *

Longueur d'un mot : nombre de lettres du mot x, noté $|x|$. Le mot ε est le mot vide, de longueur 0 .

V^* **:** ensemble des mots finis sur V

V^+ **:** ensemble des mots finis non vides sur V. On a alors $V^* = V^+ \cup \{\varepsilon\}$

Concaténation : opération binaire mettant deux mots bout à bout. On la note $\bullet$ (ou rien). Cette opération possède des propriétés :

- elle n'est pas commutative.
- elle est associative :
$$(xy)z = x(yz)$$
- elle possède un élément neutre :
$$\varepsilon : \varepsilon x = x\varepsilon = x$$
- elle est distributive pour l'opérateur $|.|$:
$$|xy| = |x| + |y|$$
- De plus, on a la propriété
$$\forall z \in V^*, zx = zy \Rightarrow x = y$$

Puissance d'un mot : La puissance se définit naturellement par récurrence

$$\begin{cases} x^0 = \varepsilon \\ \forall n \in \mathbb{N}, x^n = xx^{n-1} \end{cases}$$

Facteurs d'un mot : $x \in V^*$ est un facteur de $y \in V^*$ s'il existe $s, t \in V^*$ tels que $y = sxt$.
Si $s = \varepsilon$, $y = xt$ et x est un **préfixe** de y.
Si $t = \varepsilon$, $y = sx$ et x est un **suffixe** de y.

Ordre sur les mots : Une relation d'ordre est une relation réflexive, antisymétrique et transitive.

$$\left.\begin{array}{ll} \bullet\ \textbf{ordre préfixe} : & <_{pref}: x <_{pref} y \Leftrightarrow x \text{ préfixe de } y. \\ \bullet\ \textbf{ordre suffixe} : & <_{suff}: x <_{suff} y \Leftrightarrow x \text{ suffixe de } y. \end{array}\right\} \textit{ordre partiel}$$

- **ordre lexicographique** : c'est l'ordre du dictionnaire : *ordre total*
- **ordre produit** : $<_2$ défini sur V^2 : $(a, b) <_2 (a', b') \Leftrightarrow a \leq a'$ et $b \leq b'$.

Langage : Un langage sur un vocabulaire V est une partie de V^*, c'est à dire un ensemble de mots.
On a donc

$$L \in \mathcal{P}(V^*), L \subset V^*$$

.

Exemples Avec le vocabulaire $V = \{a, b\}$
- $L_1 = \{a, aba, bbaa, baab\}$
- $L_2 = \{ab^n a, n \in \mathbb{N}\}$
- $L_3 = \{w \in V^*, |w|_a \text{ est pair}\}$ *

2. Opérations sur les langages

Opérations ensemblistes : réunion $\bigcup$, intersection $\bigcap$

Concaténation :
- $L.L' = \{u.v/u \in L \text{ et } v \in L'\}$
- $L.\{\varepsilon\} = \{\varepsilon\}.L = L$: le neutre est ε
- $L.\varnothing = \varnothing$
- distributivité : $(K \cup L).P = K.P \cup L.P$

Puissance : On définit la fonction puissance par récurrence

$$\begin{cases} L^0 = \{\varepsilon\} \\ L^n = L.L^{n-1}, \forall n \in \mathbb{N} \end{cases}$$

Un mot de L^n est la concaténation de n mots de L qui ne sont pas forcément tous égaux.

Exemple si $L = \{a, b, c\}$, alors $L^3 = \{aaa, aab, aac, aba, abb, abc, \ldots\}$ *

Étoile de Kleene : C'est la réunion infinie de toutes les puissances d'un langage L, soit

$$L^* = \bigcup_{n \in \mathbb{N}} L^n$$

Chapitre II

Induction et Récurrence

1. *Principes*

1-a. 1^{er} principe

Théorème 1:
Soit $\mathcal{P}(n)$ un prédicat dépendant de n. Si les deux conditions suivantes sont vérifiées :
- base *: $\mathcal{P}(0)$ est vrai*
- Induction *: $\forall n \in \mathbb{N}, \mathcal{P}(n) \Rightarrow \mathcal{P}(n+1)$*

alors

$$\forall n \in \mathbb{N}, \mathcal{P}(n)$$

Exemple Montrons la propriété $\forall n \in \mathbb{N}, 8^n - 1 \equiv 0[7]$
- *base* : $8^0 - 1 \equiv 0[7]$: vrai
- *Induction* : Soit $n \in \mathbb{N}, 8^n - 1 \equiv 0[7]$.
  Alors $\exists k \in \mathbb{N}, 8^n - 1 = 7k$. D'où $8^{n+1} - 1 = 8(8^n - 1) + 7 = 8 * 7k + 7 = 7(8k+1) \equiv 0[7]$. $*$

Remarque Cette stratégie est dite **constructive** ou **ascendante** $*$

1-b. 2^{nd} principe

Théorème 2:
Soit $\mathcal{P}(n)$ un prédicat dépendant de n. Si les deux conditions suivantes sont vérifiées :
- base *: $\mathcal{P}(0)$ est vrai*
- Induction *: $\forall n \in \mathbb{N}, \prod_{i=1}^{n} \mathcal{P}(i) = \text{ vrai} \Rightarrow \mathcal{P}(n+1)$ vrai*

Alors le prédicat $\mathcal{P}(n)$ est vrai $\forall n \in \mathbb{N}$

Remarque Cette stratégie est dite **destructive** ou **descendante** $*$

Exemple Montrons que $\forall n \in \mathbb{N}, \mathcal{P}(n)$: n admet une décomposition en facteurs premiers.
- *base* : 2 est premier : vrai.
- *Induction* : Soit $n > 2$ tel que $\forall y < x, \mathcal{P}(y)$.
  - 1^{er} cas : x est premier : c'est fini !
  - 2^{eme} cas : x n'est pas premier. Alors $\exists (u, v) \in \mathbb{N}^2,\ x = uv,\ u < x$ et $v < x$.
    Par l'hypothèse d'induction, y et y' se décomposent en facteurs premiers. Ainsi, x se décompose en facteurs premiers. $*$

Attention : Ces deux principes sont équivalents pour $\mathbb{N}$. Ce n'est pas le cas en général.

2. Extension du 1^{er} principe : Induction structurelle

2-a. Introduction

Le premier principe permet d'établir le principe de démonstration par induction structurelle.

Exemple soit $X_d = \{1, 10, 100, 1000, \ldots\}$.
- $1 \in X_d$
- Si $x \in X_d, x.0 \in X_d$ $*$

On appelle U l'univers : l'ensemble de tous les éléments possibles.
Dans l'exemple ci-dessus, $U = \{0, 1\}^*$

2-b. Schéma d'induction

***Définition* 1.** *Un schéma d'induction est défini par la* base *et les* règles*. C'est-à-dire :*
- *une partie B de U : $B \subset U$:* la base*, éventuellement infinie*
- *un ensemble $\mathcal{R}$ (les* Règles*) d'opérations R :*

$$R : U^k \longrightarrow U \text{ (en nombre fini)}$$

Exemple On définit l'ensemble X_d :

$$X_d : \begin{cases} B = \{1\} \\ \mathcal{R} = \{x \longrightarrow x.0\} \end{cases}$$

L'ensemble X_d qui est défini par ce schéma d'induction vérifie les trois points :
- $B \subseteq X_d$
- application des règles : $\forall R \in \mathcal{R}, x_1, x_2, \ldots x_k \in X_d \Rightarrow R(x_1 \ldots x_k) \in X_d$
- clause de fermeture : X_d ne contient que des éléments construits en partant de la base en appliquant un nombre fini de règles.

Exemple $1000 = R(R(R(1)))$ $*$

2-c. Induction structurelle

Théorème 3:
Soit E défini par un schéma d'induction $(B, \mathcal{R})$. Soit $\mathcal{P}(x)$ un prédicat dépendant de $x \in E$.
Si les deux conditions suivantes sont vérifiées :
- Base *: $\forall x \in B \mathcal{P}(x)$ est vrai*
- Induction *: Si $\forall R : E^k \longrightarrow E \in \mathcal{R}$, on a :*

$$\forall 1 \leq i \leq k, \mathcal{P}(x_i) vrai \Rightarrow \mathcal{P}(R(x_1 \ldots x_k))$$

Alors, par le principe d'induction structurelle :

$$\forall x \in E, \mathcal{P}(x) \text{ est vrai}$$

Remarque C'est une généralisation du premier principe de récurrence dans $\mathbb{N}$. On a en fait :

$$N : \begin{cases} B = \{0\} \\ \mathcal{R} = \{R : n \longrightarrow n + 1\} \end{cases}$$

2-d. Exemple de preuve par induction structurelle

Démonstration Soit le vocabulaire $V = \{a, b\}$. Soit E défini par le schéma d'induction suivant

$$E : \begin{cases} B = \{\varepsilon\} \\ \mathcal{R} = \{R_1, R_2\} \end{cases}$$

Avec

$$\begin{cases} R_1 : (u, v) \in E^2 \longrightarrow aubv \\ R_2 : (u, v) \in E^2 \longrightarrow buav \end{cases}$$

L'idée est de montrer que

$$E = \{w \in \{a, b\}^*, |w|_a = |w|_b\}$$

On va donc faire une preuve par induction structurelle pour démontrer une inclusion. Montrons que les mots engendrés par le schéma ont bien autant de a que de b :
- *Base* : $\{\varepsilon\}$: $|\varepsilon|_a = |\varepsilon|_b = 0$: OK
- *Hypothèse d'induction* : Soient u,v deux mots engendrés par le schéma, tels que $|u|_a = |u|_b$ et $|v|_a = |v|_b$. Alors $R_1(u, v) = aubv$, et donc

  $$|R_1(u, v)|_a = |R_1(u, v)|_b$$

  De plus, $R_2(u, v) = buav$, d'où

  $$|R_2(u, v)|_a = |R_2(u, v)|_b$$

Par le principe d'induction structurelle, l'inclusion est démontrée :

$$E \subseteq L$$

On voudrait montrer l'autre inclusion. Pour cela, il faut introduire un autre type d'induction □

3. *Induction bien fondée, ou noethérienne*

3-a. Ordre bien fondé

Définition 2. *Soit E un ensemble et une relation d'ordre $\leq$. Cette relation est un* ***ordre bien fondé*** *s'il n'existe pas dans E de suite infinie strictement décroissante selon $\leq$*

Définition 3. *(Définition équivalente)*
$\leq$ est un ***ordre bien fondé*** *sur E $\Leftrightarrow$ Toute partie non vide de E admet au moins un élément minimal.*

Exemple
- $(\mathbb{Z}, <)$ n'est pas bien fondé
- $\mathcal{P}(V^*), \subset$ est bien fondé.

  Remarque Ce n'est pas un ordre total *

Propriété 1:
une condition suffisante pour qu'un ordre soit bien fondé :

$$\exists\ h : E \longrightarrow \mathbb{N}, \forall (x, y) \in E^2,\ x <_E y \Rightarrow h(x) <_{\mathbb{N}} h(y)$$

Démonstration Soit $F \subseteq E$ non vide, et soit $m = min\{h(x), x \in F\}$. On note $x_m \in F$ tel que $h(x_m) = m$. Alors

$$\forall x \in F,\ h(x_m) \leq h(x)$$

Donc x_m est un minimal de F car si il existait $z \in F$ tel que $z \leq x_m$, on aurait par définition de h : $h(z) \leq h(x_m)$, ce qui est impossible. □

3-b. Induction bien fondée

Théorème 4:
Soit $(E, <)$ bien fondé. Soit $\mathcal{P}(x)$ un prédicat dépendant de x. Si les deux conditions suivantes sont vérifiées :

1. Base *: Pour tout minimal de E, $\mathcal{P}(x)$ est vrai.*
2. Induction *:*

$$\textit{Si } \forall x \textit{ non minimal}, \forall y \in E, y < x,\ \mathcal{P}(y) \textit{ vrai } \Rightarrow \mathcal{P}(x) \textit{ est vrai}$$

Alors

$$\forall x \in E,\ \mathcal{P}(x) \textit{ est vrai.}$$

Démonstration Soit $F = \{x \in E, \mathcal{P}(x) \text{ vrai}\} \subseteq E$. Soient $R = E,\ F = \{x \in E, \mathcal{P}(x) \text{ faux}\}$.
Par l'absurde, supposons que $R \neq \varnothing$. Comme E est bien fondé, alors R admet un minimal noté r. Par définition de R, $\mathcal{P}(r)$ est faux. Donc r n'est pas un minimal de E, sinon la clause 1 s'appliquerait.
Soit $y < r$ quelconque. $y \notin R$, car sinon r ne serait pas minimal dans R. Ainsi on sait que $y \in F$. D'où $P(y)$ est vrai, et par 2., $P(r)$ est vrai.
$\Rightarrow$ Faux car $r \in R$. Ainsi, cela prouve que $R = \varnothing$ et que $E = F$. D'où le résultat :

$$\forall x \in E, \mathcal{P}(x) \text{ est vrai}$$

On a donc le résultat. □

3-c. Exemple de preuve par induction bien fondée

Exemple Application au cas du langage $L = \{w, |w|_a = |w|_b\}$. On va enfin montrer que $L \subseteq E$.

Démonstration Considérons la réunion des $L_n = \{$ mots à $2n$ lettres $\}$:

$$\bigcup_{n=0}^{+\infty} L_n$$

On veut montrer que si on prend un mot à $2n$ lettres qui a autant de a que de b, alors on peut le construire par le schéma E défini précédemment. On fait une preuve par induction bien fondée :

1. *base* : élément minimal : $\{\varepsilon\}$: $|\varepsilon|_a = |\varepsilon|_b = 0$. ε est engendré par E car c'est la base du schéma.
2. *induction* : Soit $n \in \mathbb{N}$ tel que l'on ait :

$$\forall i < n, L_i \text{ est engendré par } E$$

Montrons que L_n est engendrée par E.
Soit $x \in L_n$. Supposons par exemple que x commence par la lettre a. Alors :

$$\exists (u, v) \in L, |u|_a = |u|_b, |v|_a = |v|_b, x = aubv$$

De plus,

$$|x| = 2n \Rightarrow |u|, |v| \leq 2(n-1)$$
$$\Rightarrow \exists (i, j) \in \mathbb{N}^2, u \in L_i, v \in L_j$$

Par l'hypothèse d'induction, u et v sont générés par E. Ainsi, en appliquant $u, v \longrightarrow x = aubv$, on sait que x est engendré par E. On a donc montré que :

$$\forall x \in L,\ x \text{ est engendré par } E$$

Exemple Le *pgcd*.

$$\forall a \in \mathbb{N}^*,\ pgcd(a, a) = a$$

$$\forall (a, b) \in \mathbb{N}^{*^2},\ pgcd(a, b) = \begin{cases} pgcd(a, b-a) \text{ si } a < b \\ pgcd(a-b, b) \text{ si } a > b \end{cases}$$

Cette fonction est-elle bien définie ? Sait-on le calculer pour toute valeur de $\mathbb{N}^* \times \mathbb{N}^*$?

DÉMONSTRATION On va déjà montrer que l'on a une relation d'ordre bien fondée.

1. *Ordre bien fondé.*

$$(a,b) <_{app} \left\{ \begin{array}{l} (a+b,b) \\ (a,a+b) \end{array} \right.$$

Cet ordre est bien fondé car $h(a,b) = a+b$ satisfait :

$$(a,b) <_{app} (a',b') \Rightarrow h(a,b) < h(a',b')$$

On peut donc maintenant montrer que *pgcd* est bien définie.

2. *Base.* éléments minimaux $= \{(a,a), a \in \mathbb{N}\}$ et $pgcd(a,a)$ est bien défini.
3. *Induction.* Soit un couple (a,b) non minimal. On suppose que $\forall (x,y) <_{app} (a,b)$, $pgcd(x,y)$ existe. Alors :

$$pgcd(a,b) = \left\{ \begin{array}{l} pgcd(a,b-a) \text{ si } a < b \text{ et } (a,b-a) <_{app} (a,b) \\ pgcd(a-b,b) \text{ si } a > b \text{ et } (a-b,b) <_{app} (a,b) \end{array} \right.$$

donc $pgcd(a,b)$ est défini. □

Exemple Fonction de Ackerman : elle est définie par récurrence :

$$\forall (m,n) \in \mathbb{N}^2, \left\{ \begin{array}{l} A(0,n) = n+1 \\ A(m,0) = A(m-1,1) \\ A(m,n) = A(m-1, A(m,n-1)) \end{array} \right.$$

On peut la plonger dans l'ordre lexicographique des couples :

$$(x,y) < (x',y') \Leftrightarrow x < x' \text{ ou } x = x' \text{ et } y < y'$$

4. Fonctions définies sur un schéma d'induction

Exemple Soit la fonction $f : \mathbb{N} \longrightarrow \mathbb{N}$ définie par :

$$\left\{ \begin{array}{l} f(0) = 1 \\ f(n+1) = (n+1)f(n) \end{array} \right.$$

Alors cette fonction peut être définie sur $\mathbb{N}$ par le schéma d'induction suivant :

$$\left\{ \begin{array}{l} \text{Base : } 0 \\ \text{Règles : } n \longrightarrow n+1 \end{array} \right.$$

Définition 4. *Soit E définie sur un schéma d'induction*

$$\left\{ \begin{array}{l} \text{Base : } \mathcal{B} \subset E \\ \text{Règles : } \mathcal{R} : U^k \longrightarrow U \end{array} \right.$$

On peut définir une fonction f :

$$\left\{ \begin{array}{l} \text{Base : } \forall x \in \mathcal{B}, f(x) \\ \text{Règles : } \forall \mathcal{R}, f(\mathcal{R}(x_1, x_2, \ldots, x_n)) \end{array} \right.$$

Exemple $m : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$

$$\left\{ \begin{array}{l} \text{Base : } m(a,0) = 0 \\ \text{Règles : } m(a,n+1) = m(a,n) + a \end{array} \right.$$

$$\Rightarrow m(a,n) = na$$

Exemple $m : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$

$$\left\{ \begin{array}{l} \text{Base : } m(a,0) = 0 \\ \text{Règles : } \begin{array}{lcl} m(a,2n) & = & m(2a,n) \\ m(a,2n+1) & = & a + m(2a,n) \end{array} \end{array} \right.$$

$$\Rightarrow m(a,n) = na$$

Exemple $m : V^* \longrightarrow \mathbb{N}$

$$\left\{ \begin{array}{l} m(\varepsilon) = 0 \\ \forall a \in V, m(a.u) = 1 + m(a) \end{array} \right.$$

$$\Rightarrow \text{ Longueur du mot}$$

Exemple $X_d = \{1, 10, 100, 1000, \ldots\}$. On a en fait :

$$X_d : \left\{ \begin{array}{l} \mathcal{B} = \{1\} \\ \mathcal{R} : x \longrightarrow x.0 \end{array} \right.$$

on pose $l : \left\{ \begin{array}{l} l(1) = 0 \\ l(x.0) = 1 + l(x) \end{array} \right.$ et $p : \left\{ \begin{array}{l} p(1) = 1 \\ p(x.0) = 2p(x) \end{array} \right.$.
Alors on a la propriété suivante :

$$p(x) = 2^{l(x)}$$

.

Démonstration On peut montrer cela par induction structurelle.

- *base* : $p(1) = 1$ et $2^{l(1)} = 2^0 = 1$
- *induction* : Soit $x \in X_d$ tel que $p(x) = 2^{l(x)}$.

$$\begin{array}{lcl} p(x.0) & = & 2p(x) \\ & = & 2.2^{l(x)} \\ & = & 2.2^{l(x.0)-1} \\ & = & 2^{l(x.0)} \end{array}$$

Exemple

$$V^* : \left\{ \begin{array}{l} \text{Base : } : \varepsilon \\ \text{Induction : } \mathcal{R} : \forall a \in V, y \longrightarrow ay \end{array} \right.$$

On peut alors définir sur V^* l'**ordre lexicographique** : $f(x,y) \Leftrightarrow x < y$:

$$\left\{ \begin{array}{l} \text{Base : } f(\varepsilon, x) \textit{ est vrai} \\ \text{Induction : } f(ay, x) \textit{ est vrai} \Leftrightarrow (x = ax' \textit{ et } f(y,x) \textit{ est vrai}) \textit{ ou } (x = bx' \textit{ et } f(a,b) \textit{ est vrai}) \end{array} \right.$$

. *

Chapitre III

Grammaires

1. Définition d'une grammaire

Exemple

$$X_d : \left[\begin{array}{l} \mathcal{S} \longrightarrow 1 \\ \mathcal{S} \longrightarrow \mathcal{S}0 \end{array} \right.$$

On peut former des mots de cette façon :

$$\mathcal{S} \Rightarrow^1 1$$

$$\mathcal{S} \Rightarrow^2 \mathcal{S}0 \Rightarrow^2 \mathcal{S}00 \Rightarrow^1 100$$

Définition 5. *Une* ***grammaire*** $G =< V_T, V_N, \mathcal{S}, \mathcal{R} >$

V_T	:	*Vocabulaire terminal*	*(le V de d'habitude)*
V_N	:	*Vocabulaire non terminal*	*(symboles auxiliaires)*
$\mathcal{S} \in V_N$	:	*Axiome*	*(Start)*
$\mathcal{R}$	:	*Règles de réécriture*	

Exemple $G =< \{a,b\}, \{\mathcal{S}\}, \mathcal{S}, \mathcal{S} \longrightarrow \varepsilon \mid a\mathcal{S}b\mathcal{S} \mid b\mathcal{S}a\mathcal{S}\} >$ On a alors :

$$L(G) = \{m \in \{a,b\}^*, |m|_a = |m|_b\}$$

2. Relation de dérivation

Définition 6. *On dit que $x \Longrightarrow y$ si et seulement s'il existe une rège $u \longrightarrow v$ telle que*

$$\left\{ \begin{array}{l} x = w_1 u w_2 \\ y = w_1 v w_2 \end{array} \right.$$

Exemple Avec $A \longrightarrow ab \in \mathcal{R}$.
Si $x = cAc$ et $y = cabc$, alors $x \Longrightarrow y$. *

Définition 7. ***Dérivation*** *: Suite finie de mots qui dérivent les uns des autres.*

$$x_0 \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_k$$

x_k est dérivé à partir de x_0 en k pas. On le note encore

$$x_0 \Longrightarrow^k x_k$$

On définit classiquement $\Longrightarrow^*$ et $\Longrightarrow^+$.

Exemple On peut donc écrire pour X_d :

$$\mathcal{S} \Longrightarrow^* 1000 \textit{ et } \mathcal{S} \Longrightarrow^4 1000$$

Théorème 5:
Le langage engendré par une grammaire G est :

$$L(G) = \{x \in V_T^*, S \Rightarrow^* x\}$$

Exemple $G =< \{a, b\}, \{\mathcal{S}\}, \mathcal{S}, \{\mathcal{S} \longrightarrow a\mathcal{S} \mid b\mathcal{S} \mid \varepsilon\} >$
On voit clairement que $L(G) =$ tous les mots composés de a et de b. *

Exemple [**Langage des parenthèses**]
On pose $V_T = \{a, b\}$ et $V_N = \{\mathcal{S}\}$. Le langage des parenthèses peut être défini par ces trois grammaires :

$$\begin{array}{ll} G_1 : & \mathcal{S} \longrightarrow a\mathcal{S}b \mid \mathcal{S}\mathcal{S} \mid \varepsilon \\ G_2 : & \mathcal{S} \longrightarrow a\mathcal{S}b\mathcal{S} \mid \varepsilon \\ G_3 : & \mathcal{S} \longrightarrow \mathcal{S}a\mathcal{S}b \mid \varepsilon \end{array}$$

Exemple $G' : \mathcal{S} \longrightarrow a\mathcal{S} \mid b\mathcal{S} \mid \mathcal{S}a \mid \mathcal{S}b \mid \varepsilon$: **Grammaire ambigue** En effet, on peut obtenir un même mot en appliquant des règles différentes :

$$\mathcal{S} \rightarrow a\mathcal{S} \rightarrow aa\mathcal{S} \rightarrow aa$$

$$\mathcal{S} \rightarrow \mathcal{S}a \rightarrow a\mathcal{S}a \rightarrow aa$$

An contraire, $G =< \{a, b\}, \{\mathcal{S}\}, \mathcal{S}, \{\mathcal{S} \longrightarrow a\mathcal{S} \mid b\mathcal{S} \mid \varepsilon\} >$ n'est pas ambigue *

Exemple Le langage des mots sur $\{a, b\}$ qui ont un nombre pair de a peut être défini ainsi :

$$\left\{ \begin{array}{l} \mathcal{S} \longrightarrow b\mathcal{S} \mid a\mathcal{T} \mid \varepsilon \\ \mathcal{T} \longrightarrow a\mathcal{S} \mid b\mathcal{T} \end{array} \right.$$

$\Rightarrow$ Linéraire à droite. *

Exemple

$$\left\{ \begin{array}{l} \mathcal{S} \longrightarrow \#a\# \\ \#a \longrightarrow \#B \\ Ba \longrightarrow aaB \\ B\# \longrightarrow aa\# \end{array} \right. \Rightarrow \text{ quelconque}$$

$$\rightarrow \#aaaa\# \rightarrow \#Baaa\# \rightarrow \#aaBaa\#$$

Langage	Grammaires	Exemples	$x \in L(G)$	$L(G_1) = L(G_2)$	Ambigu
Réguliers Rationnels	**Linéaires à droite** $\left[\begin{array}{l} A \to uB \\ A \to u \end{array}\right.$ $A, B \in V_n$ $u \in V_T^*$	Nombre pair de a V^* $X_d = \{1, 10, 100 \ldots\}$	oui	oui	oui
Hors Contexte	**Hors contexte** $A \to \alpha, A \in V_N$ et $\alpha \in (V_N \cup V_T)^*$	$\{a^n b^n / n > 0\}$ palindrômes $\omega\tilde{\omega}$	oui	non	non
Quelconques, Récursivement énumérables	**Quelconques** $\alpha \to \beta$ $(\alpha, \beta) \in (V_N \cup V_T)^*$	$\#a^{2^n}\#$ $\{a^n b^n c^n / n \geq 0\}$	non	non	non
Non récursivement énumérable	**Inexistantes**	{ programmes Ada qui calculent le *pgcd* }	non	non	non

Chapitre IV

Les Langages Réguliers

1. Expressions régulières

Définition 8. *Soit V un vocabulaire. On définit le schéma d'induction E suivant :*
- *Base : ε et $\varnothing \in E$, et $\forall a \in V, a \in E$*
- *Règles :*

$$\left[\begin{array}{l} e_1, e_2 \longrightarrow e_1 + e_2 \\ e_1, e_2 \longrightarrow e_1.e_2 \\ e \longrightarrow e^* \end{array}\right.$$

Définition 9. [Langage représenté par une expression régulière]
1. *Base :*

$$L(\varepsilon) = \{\varepsilon\}$$
$$L(\varnothing) = \{\varnothing\}$$
$$L(a) = \{a\}$$

2. *Règles :*

$$L(e_1 + e_2) = L(e_1) \cup L(e_2)$$
$$L(e_1.e_2) = L(e_1).L(e_2)$$
$$L(e^*) = L(e)^*$$

Exemple Si $E = ((a^* + b.c)^* + \varepsilon) + \varnothing$, alors on a :

$$L(E) = (\{a\}^* \cup \{bc\})^* \cup \{\varepsilon\} \cup \{\varnothing\}$$

Exemple $10^* = X_d$ *

Exemple On peut comparer des expressions régulières pour savoir si elles sont égales :
- $(a^*(ab)^*)^* = (a + ab)^*$
- $(a^*b)^* \neq (a + b)^* = V^*$ car $(a^*b)^*$ ne contient pas le mot a. *

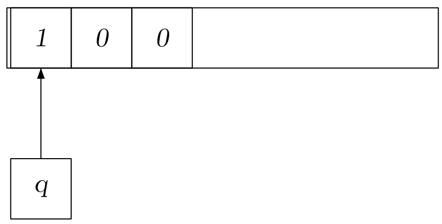
Exemple Sous Unix, la commande `grep -E '(ât|âmes|ass(e|es|ion|iez|ent))' toto.txt` utilise en fait l'expression régulière suivante :

at + âmes + ass.(e+es+ions+iez+ent) *

D'autres commandes font aussi appels à des expressions régulières, comme `sed` ou encore `perl`.
Dans les langages C, C++, Ada, elles sont utilisées dans la librairie RegExp.

2. Automates Finis

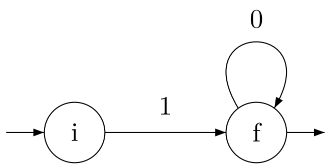
***Définition* 10.** MACHINE ABSTRAITE À MÉMOIRE FINIE



Formellement, on définit une machine par $\mathcal{M} = \langle Q, V, \delta, i, F\rangle$ *:*

Q	:	*Ensemble fini d'états*	
V	:	*Vocabulaire*	
δ	:	*Ensemble des transitions*	$\delta \in \mathcal{P}(Q \times V \cup \{\varepsilon\} \times Q)$
i	:	*État initial*	$i \in Q$
F	:	*États finaux*	$F \subset Q$

Exemple X_d, le retour ...

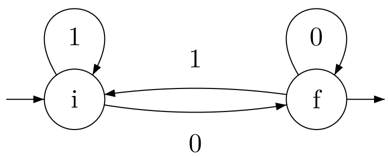


on peut facilement voir avec cet automate si un mot appartient ou non à X_d.

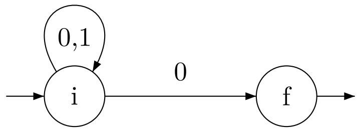
$$\begin{array}{lll} 100: & i \xrightarrow{1} f \xrightarrow{0} f \xrightarrow{0} f & \Rightarrow \quad 100 \in X_d \\ 101: & i \xrightarrow{1} f \xrightarrow{0} f \xrightarrow{1} \times & \Rightarrow \quad 101 \notin X_d \end{array}$$

*

Exemple Constantes binaires paires



On a aussi une autre solution, mais avec un automate qui n'est pas déterministe :



On peut donner la table de transition de ce dernier automate :

	0	1
i	i, f	i
f	$\varnothing$	$\varnothing$

*

Définition 11. Langage reconnu par un automate
Le langage reconnu par un automate est l'ensemble des mots tel qu'il existe un chemin dans l'automate qui part de l'état initial, qui aboutit à un état final, et dont les transitions portent les lettres du mot. Formellement, le mot $\omega = a_1 a_2 \dots a_n \in L(\mathcal{M})$ si et seulement si

$$\exists (q_0, q_1, \dots, q_n), \begin{cases} q_0 = i \\ q_n \in F \\ \forall i \in 1..n, (q_{i-1}, a_i, q_i) \in \delta \end{cases}$$

Remarque Il y a des modèles équivalents, avec :
- plusieurs états initiaux
- un seul état final
- des mots comme transition. *

Exemple $a^* b^*$

*

Pour ce même langage, on peut donner la grammaire associée :

$$\left[\begin{array}{l} \mathcal{S} \longrightarrow a\mathcal{S} \mid \mathcal{T} \\ \mathcal{T} \longrightarrow b\mathcal{T} \mid \varepsilon \end{array} \right. \quad \text{ou} \quad \left[\begin{array}{l} \mathcal{S} \longrightarrow \mathcal{S}a \mid \mathcal{T} \\ \mathcal{T} \longrightarrow \mathcal{T}b \mid \varepsilon \end{array} \right.$$

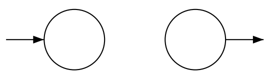
3. Équivalence des trois modèles

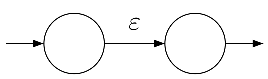
3-a. Lien entre langage régulier et automate fini

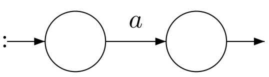
Théorème 6:
Tout langage régulier est reconnu par un automate fini.

Démonstration On fait bien entendu une preuve par induction structurelle.

1. **Base** :

   (a) $\varnothing$: 

   (b) ε : 

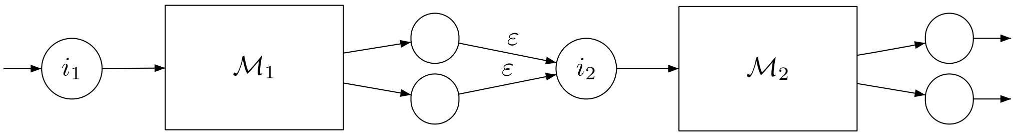
   (c) $a \in V$: 

2. **Induction** : Si on a e_1 et e_2 reconnus respectivement par les automates

$$\mathcal{M}_1 = \langle Q_1, V_1, \delta_1, i_1, F_1 \rangle$$

$$\mathcal{M}_2 = \langle Q_2, V_2, \delta_2, i_2, F_2 \rangle$$

(a) e_1, e_2 sont reconnus $\Rightarrow e_1.e_2$ est reconnu par un automate fini.

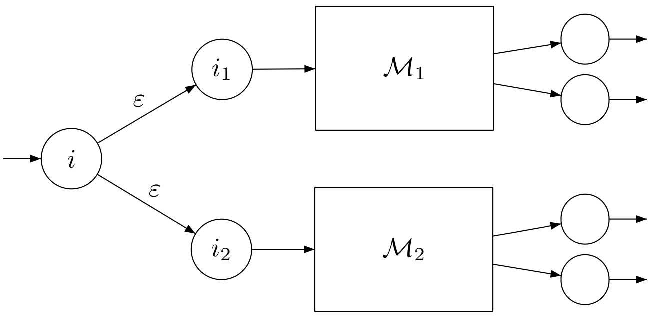


On en déduit par l'automate ci-dessus que $e_1.e_2$ est reconnu par l'automate :

$$\mathcal{M}_{\bullet} = \langle Q_1 \cup Q_2, V_1 \cup V_2, \delta_{\bullet}, i_1, F_2 \rangle$$

$$\text{avec } \delta_{\bullet} = \delta_1 \cup \delta_2 \cup F_1 \times \{\varepsilon\} \times \{i_2\}$$

(b) e_1, e_2 sont reconnus $\Rightarrow e_1 + e_2$ est reconnu par un automate fini.

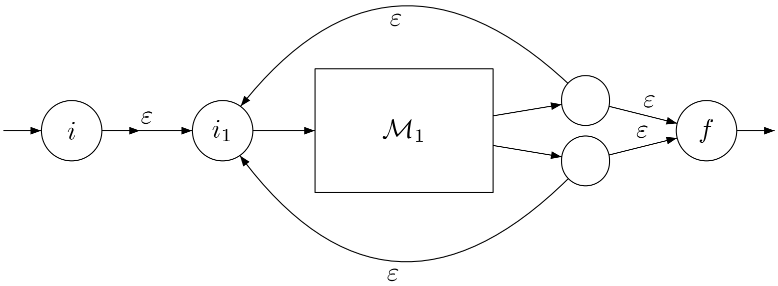


On en déduit par l'automate ci-dessus que $e_1 + e_2$ est reconnu par l'automate :

$$\mathcal{M}_{+} = \langle Q_1 \cup Q_2 \cup \{i\}, V_1 \cup V_2, \delta_{+}, i, F_1 \cup F_2 \rangle$$

$$\text{avec } \delta_{+} = \delta_1 \cup \delta_2 \cup \{(i, \varepsilon, i_1), (i, \varepsilon, i_2)\}$$

(c) e_1 est reconnu $\Rightarrow e_1^*$ est reconnu par un automate fini.



□

Le langage régulier e_1^* est reconnu par l'automate :

$$\mathcal{M}_{*} = \langle Q_1 \cup \{i, f\}, V_1, \delta_{*}, i, \{i, f\} \rangle$$

$$\text{avec } \delta_{*} = \delta_1 \cup \{(q, \varepsilon, f)/q \in F_1\} \cup \{(q, \varepsilon, i_1)/q \in F_1\} \cup \{(i, \varepsilon, i_1)\}$$

3-b. Langage reconnu par un automate

Propriété 2:
Soit l'automate $\mathcal{M}$ défini par :

$$\mathcal{M} = \langle Q, V \cup \{\varepsilon\}, \delta, i, F \rangle$$

$$Q = \{q_1, q_2, \dots, q_n\}$$

$$\alpha_{i,j} = \{a \in V \cup \{\varepsilon\}, (q_i, a, q_j) \in \delta\}$$

On peut donner le système d'équation associée à $\mathcal{M}$:

$$\begin{cases} x_i = \varepsilon + \displaystyle\sum_{j=1}^{n} \alpha_{i,j} x_j \ \textit{si}\ q_i \in F \\ x_i = \displaystyle\sum_{j=1}^{n} \alpha_{i,j} x_j \ \textit{sinon} \end{cases}$$

x_i est un langage. C'est le langage reconnu par $\mathcal{M}$ partant de l'état q_i. Le système a pour solution :

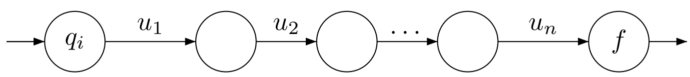
$$\begin{array}{rcl} x_1 & = & L(\mathcal{M}_1) \\ x_2 & = & L(\mathcal{M}_2) \\ & \vdots & \\ x_n & = & L(\mathcal{M}_n) \end{array}$$

avec

$$\mathcal{M}_i = \langle Q, V \cup \{\varepsilon\}, \delta, q_i, F \rangle$$

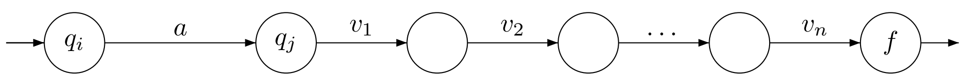
Démonstration Soit $u = u_1 u_2 \dots u_n \in V^*$. Alors :

$u \in L(\mathcal{M}_i) \Leftrightarrow$ il existe un chemin partant de q_i, arrivant à un état final et étiqueté par u.



C'est encore équivalent à :
- soit $u \in \varepsilon$ et $q_i \in F$
- soit il existe un chemin de longueur strictement positive entre q_i et un état final, c'est à dire :

$$u = av \ \textit{avec}\ a \in \alpha_{i,j}, v \in L(\mathcal{M}_j)$$



$$\Leftrightarrow \begin{cases} L(\mathcal{M}_i) = \varepsilon + \displaystyle\sum_{j=1}^{n} \alpha_{i,j} L(\mathcal{M}_j) \ \textit{si}\ q_i \in F \\ x_i = \displaystyle\sum_{j=1}^{n} \alpha_{i,j} L(\mathcal{M}_j) \ \textit{sinon} \end{cases}$$

Ainsi, le langage reconnu par $\mathcal{M}$ est $L(\mathcal{M}_k)$ avec $q_k = i$ □

3-c. Démonstrations sur les langages réguliers

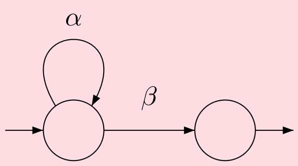
Exemple

$$\begin{cases} x_1 = 1.x_2 \\ x_2 = \varepsilon + 0.x_2 \end{cases}$$

On cherche à obtenir x_2 à partir de l'équation $x_2 = \varepsilon + 0.x_2$ On va donc introduire le thèorème suivant. $*$

Théorème 7 (Lemme d'Arden):
Les équations de la forme $x = \alpha x + \beta$ ont une plus petite solution qui vaut :

$$x = \alpha^* \beta$$



Démonstration On montre que le langage $x = \alpha^*\beta$ est la plus petite solution :

1. $\alpha^*\beta$ est une solution.

$$\begin{aligned} \alpha\alpha^*\beta + \beta &= (\alpha\alpha^* + \varepsilon)\beta \\ &= (\alpha^+ + \varepsilon)\beta \\ &= \alpha^*\beta \end{aligned}$$

2. C'est la plus petite au sens de l'inclusion. Soit L solution de $\alpha x + \beta = x$.

$$\alpha^*\beta = \bigcup_{n \geq 0} \alpha_i \beta$$

Montrons par récurrence sur i que $\forall i, \alpha^i\beta \subseteq L$:

(a) Base : $\alpha^0\beta = \varepsilon\beta = \beta$ et $\beta \subseteq L$ car $L = \alpha L + \beta$

(b) Hypothèse d'induction : $\alpha^i\beta \subseteq L$

(c) Induction : $\alpha^{i+1}\beta = \alpha\alpha^i\beta \subseteq \alpha L \subseteq \alpha L + \beta = L$ □

Exemple Si on a l'équation :

$$x_1 = (x_2 + a)x_1 + x_2.x_4$$

Alors la solution pour x_1 est :

$$x_1 = (x_2 + a)^*(x_2.x_4)$$

Théorème 8:
Expression régulière ⇔ Automates finis ⇔ Grammaires linéaires à droite

Démonstration On a déjà vu l'équivalence entre expression régulière et automate fini. On constate également l'équivalence avec les grammaires linéaires droites, en considérant la grammaire suivante :

$$\begin{cases} \mathcal{S}_i \longrightarrow \sum_{j=1}^{n} \alpha_{i,j}\mathcal{S}_j \\ \mathcal{S}_i \longrightarrow \sum_{j=1}^{n} \alpha_{i,j}\mathcal{S}_j | \varepsilon \text{ } si \text{ } q_i \in F \end{cases}$$

Exemple

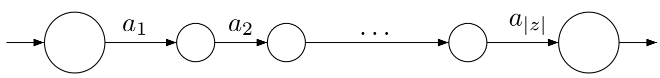
$$\left\{\begin{array}{l} A \longrightarrow u_1 B \\ A \longrightarrow u_2 \end{array}\right. \Longrightarrow x_A = u_1 x_B + u_2$$

Exemple Le langage $\{a^n b^n, n \geq 0\}$ n'est pas un langage régulier. $*$

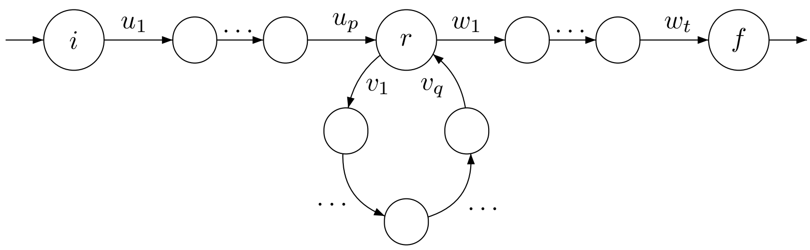
Théorème 9 (Lemme de l'étoile, Lemme de la pompe):
Soit L un langage régulier. Alors il existe $p \in \mathbb{N}$ tel que tout mot $z \in L, |z| > p$ peut s'écrire $z = uvw$, tel que :

1. $v \neq \varepsilon$
2. $|uv| \leq p$
3. $\forall n \in \mathbb{N}, uv^n w \in L$

Démonstration L est régulier donc est reconnaissable par un automate d'états finis, noté $L = L(A)$. Soit p le nombre d'états de A. On suppose de plus que cet automate n'est pas à $\varepsilon-$transitions.
Soit $z \in L$ tel que z soit reconnu par A, et $|z| \geq p$. On peut noter $z = a_1 a_2 \ldots a_{|z|}$.



$|z| > p$ donc le chemin est de longueur supérieure à p. Donc ce chemin passe au moins deux fois par le même état. Soit r un de ces états.



On note :
- $u = u_1 \ldots u_p$ le mot entre i et r
- $v = v_1 \ldots v_q$ le mot entre r et r
- $w = w_1 \ldots w_t$ le mot entre r et l'état final.

On a bien $z = uvw$. De plus, comme on peut faire autant de tour de boucle que l'on veut :

$$\forall n \in \mathbb{N}, uv^n w \in L$$

1. $v \neq \varepsilon$ car il faut une boucle.
2. $|uv| \leq p$: on choisit le r le plus proche de l'état i. Autrement dit, on prend u et v les plus petits possibles. Ainsi, le chemin qui reconnait uv ne passe jamais deux fois par le même état (sauf r à la fin).

Remarque C'est une **condition suffisante** : il y a des langages non réguliers qui satisfont **aussi** ce lemme. $*$

Remarque Pour prouver qu'un langage n'est pas régulier, on utilise la contraposée du lemme :

$\forall p,\ \exists z \in L, |z| > p$, tel que pour toute décomposition $z = uvw$ avec $v \neq \varepsilon$ et $|uv| \leq p$. On trouve un n tel que $uv^n w \notin L$. $*$

Exemple $a^n b^n, n \geq 0$ n'est pas régulier. En effet :
Soit p quelconque. Prenons $z = a^{p+1}b^{p+1}$. On a donc $|z| > p$. On analyse toutes les décomposisions possibles. $z = uvw$ avec $v \neq \varepsilon$ et $|uv| \leq p$. On a en fait un seul cas :

$$v = a^i, 0 < i \leq p+1$$

Or pour $n = 0$, $uv^n w = uv^0 w = a^{p+1-i}b^{p+1} \notin L$. Le lemme ne peut pas s'appliquer : le langage n'est donc pas régulier. *

4. Automates déterministes

Définition 12. *Soit l'automate $A = \langle Q, V, \delta, i, F\rangle$ Alors l'automate A est* ***déterministe*** *si et seulement si il n'a pas d'$\varepsilon-$ transitions et :*

$$\forall q \in Q, \forall a \in V, \text{ il y a au plus un } p \in Q, (q, a, p) \in \delta$$

Si on a en plus :

$$\exists!\ p \in Q, (q, a, p) \in \delta$$

Alors l'automate est ***complet****.*

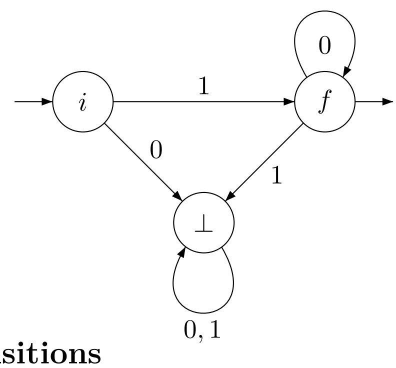
Remarque On peut dire que δ est une fonction et écrire $\delta(q, a) = p$. On peut étendre δ à :

$$\begin{cases} \delta(q, \varepsilon) = q \\ \forall a \in V, \forall x \in V^*, \delta(q, ax) = \delta(\delta(q, a), x) \end{cases}$$

$\delta(q, x) = p \Leftrightarrow$ il existe un chemin entre q et p étiqueté par x. Le chemin est de longueur $|x|$, et donc :

$$L(A) = \{x \in V^*, \delta(i, x) \in F\}$$

Exemple L'automate suivant, qui a pour langage X_d, est complet et déterministe :



4-a. Élimination des ε-transitions

Théorème 10:
Si un langage L sur V est reconnu par un automate fini, alors il est reconnu par un automate fini sans ε-transitions.

Démonstration Soit $A = \langle Q, V \cup \{\varepsilon\}, \delta, i, F\rangle$ un automate quelconque. On construit l'automate $B = \langle Q, V, \eta, i, G\rangle$ avec :

$$(p, a, q) \in \eta \Leftrightarrow \exists r \in Q \text{ tel qu'un chemin de } A \text{ étiqueté } \varepsilon \text{ va de } p \text{ à } r \text{ et } (r, a, q) \in \delta.$$

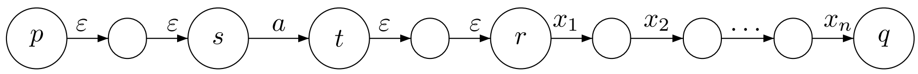
$$G = F \cup \{p \in Q, \text{ un chemin de } A \text{ va de } p \text{ à un état final et est étiqueté par } \varepsilon\}$$

Prouvons que A et B sont équivalents.

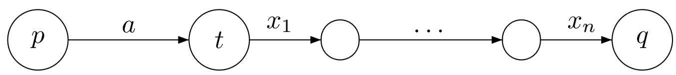
1. On va déjà montrer la propriété :

   Propriété 3:
   $P(x)$: $\exists$ un chemin de A étiqueté x entre p et $q \Leftrightarrow \exists r \in Q$, tel qu'il existe un chemin de B étiqueté x de p à r et un chemin de A étiqueté ε menant de r à q.

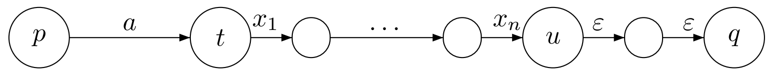
   DÉMONSTRATION On démontre cette propriété par récurrence sur $|x|$:
   – **base** : $|x| = 0 : x = \varepsilon$. Alors $P(x)$ est vrai avec $p = r$.
   – **Induction** Si $P(x)$ est vrai pour $|x| = n$:
   On considère le mot $ax, a \in V$. Il existe un chemin étiqueté ax dans A entre p et q.
   $\Leftrightarrow \exists r \in Q$, tel qu' il y a dans A un chemin étiqueté a de p à r et un chemin étiqueté x de r à q.
   $\Leftrightarrow \exists r, s, t \in Q$ tels qu'un chemin de A étiqueté ε va de p à s et de t à r et $(s, a, t) \in \delta$, et un chemin étiqueté x va de r à q.



   $\Leftrightarrow$ par définition de B : $\exists (t, r) \in Q^2, (p, a, t) \in \eta$ et $\exists$ un chemin dans A étiqueté x entre r et q et un chemin étiqueté ε entre t et r.
   $\Leftrightarrow \exists t \in Q, (p, a, t) \in \eta$ et $\exists$ un chemin dans A étiqueté x entre t et q



   $\Leftrightarrow$ Par l'hypothèse de récurrence, il existe $u \in Q$ tel que :



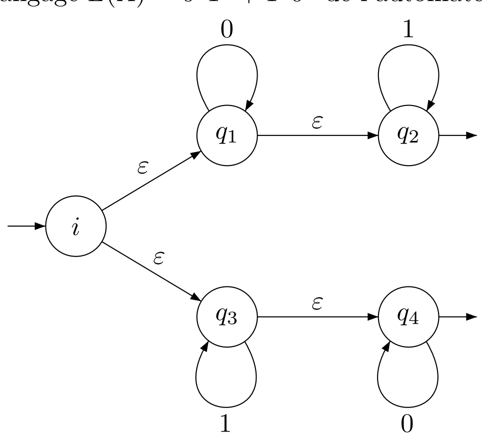
   $\Leftrightarrow P(ax)$. □

2. Montrons que $L(A) = L(B)$.

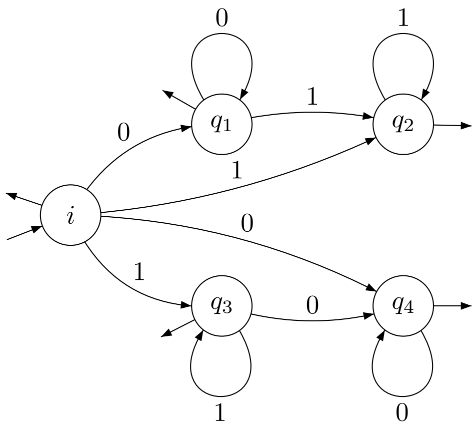
$$\begin{array}{rcl} x \in L(A) & \Leftrightarrow & \exists \text{ un chemin dans } A \text{ entre } i \text{ et } f \in F \text{ étiqueté par } x \\ \text{par } P(x) & \Leftrightarrow & \exists\, r \in Q \text{ tel qu'un chemin de } B \text{ va de } i \text{ à } r \text{ et un chemin étiqueté } \varepsilon \\ & & \text{de } A \text{ va de } r \text{ à } f \\ & \Leftrightarrow & \text{Par définition de } G, \text{ il existe dans } B \text{ un chemin de } i \text{ à } r \in G \\ & & \text{étiqueté } x \\ & \Leftrightarrow & x \in L(B) \end{array}$$

□

Exemple On considère le langage $L(A) = 0^*1^* + 1^*0^*$ de l'automate suivant :



En utilisant la méthode décrite précédemment, on peut construire l'automate B correspondant, sans ε-transitions.



$$f(p) = \{q \in Q, (p, \varepsilon, q) \in \eta\} \tag{IV.1}$$

$$g(p, x) = \bigcup_{r \in f(p)} \{q \in Q, (r, a, q) \in \delta\} \tag{IV.2}$$

*

$$r \in g(p, x) \Leftrightarrow (p, a, r) \in \eta$$

4-b. Equivalence entre automates et automates déterministes

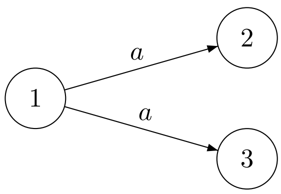
Théorème 11:
Tout langage reconnu par un automate non déterministe sans ε-transitions peut être reconnu par un automate déterministe.

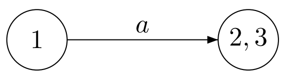
Démonstration Soit $A = \langle Q, V, \delta, i, F \rangle$ sans ε-transitions.
On construit l'automate B défini par : $B = \langle \mathcal{P}(Q), V, \eta, i, G \rangle$ avec :

$$\eta(P, a) = \bigcup_{p \in P} \{q \in Q, (p, a, q) \in \delta\}$$

$$P \in \mathcal{P}(Q)$$

$$G = \{P \in \mathcal{P}(Q), P \cup F \neq \varnothing\}$$

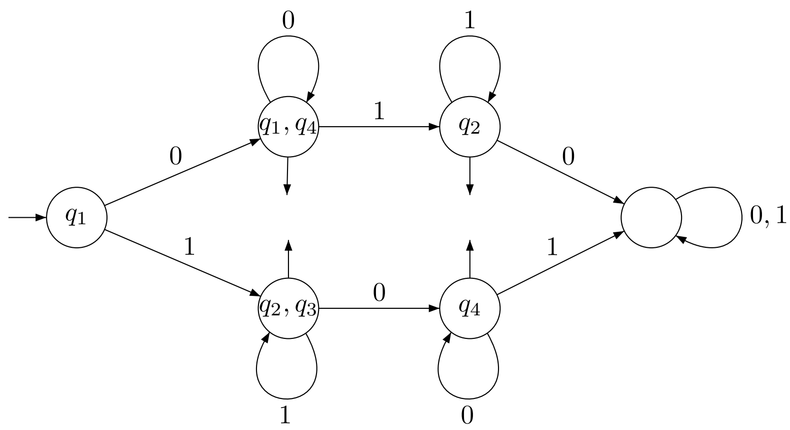




Exemple $0^*1^* + 1^*0^*$

	0	1
q_0	q_1, q_4	q_1, q_3
q_1	q_1	q_2
q_2	$\varnothing$	q_2
q_3	q_4	q_3
q_4	q_4	$\varnothing$
$\varnothing$	$\varnothing$	$\varnothing$

*



	0	1
q_0	$\{q_1, q_4\}$	$\{q_2, q_3\}$
q_1, q_4	$\{q_1, q_4\}$	$\{q_2\}$
q_2, q_3	$\{q_4\}$	$\{q_2, q_3\}$
q_2	$\varnothing$	q_2
q_4	q_4	$\varnothing$
$\varnothing$	$\varnothing$	$\varnothing$

Suite de la preuve :
Montrons maintenant que $L(A) = L(B)$:

1. $\eta(P, x) = \bigcup_{p \in P} \{q \in Q, \text{ il y a un chemin étiqueté } x \text{ dans } A \text{ qui va de } p \text{ à } q\}$
2.

$$\begin{aligned} x \in L(A) &\Leftrightarrow \text{ Il existe un chemin étiqueté } x \text{ dans } A \text{ entre } i \text{ et un état final de } A \\ &\Leftrightarrow \eta(\{i\}, x) \cup F \neq \varnothing \\ &\Leftrightarrow \eta(\{i\}, x) \in G \\ &\Leftrightarrow x \in L(B) \end{aligned}$$

Démonstration Preuve de 1 On fait une récurrence sur $|x|$:

1. **Base** : $x = \varepsilon$. A est sans ε-transitions donc
   - $P = \bigcup_{p \in P} \{q \in Q, \exists \text{ chemin de trace } \varepsilon \text{ entre } p \text{ et } q.\ \}$
   - $\eta(P, \varepsilon) = P$ car B est déterministe. □

   Ainsi, 1. = 2..
2. **Induction** : Supposons que 1. est vrai pour $x \in V^+$. On montre que c'est encore vrai pour $xa, a \in V$.
   $s \in \bigcup_{p \in P} \{q \in Q, \text{ il existe un chemin étiqueté } xa \text{ entre } p \text{ et } q\}$
   $\Leftrightarrow \exists r \in P, \text{ il existe un chemin étiqueté } xa \text{ entre } r \text{ et } s$
   $\Leftrightarrow \exists r \in P, \exists t \in Q, x$ est un chemin entre r et t et un chemin étiqueté a entre t et s
   $\Leftrightarrow$ il existe un chemin étiqueté x entre r et t
   $\Leftrightarrow t \in \bigcup_{p \in P} \{q \in Q, \text{ il existe un chemin étiqueté } x \text{ entre } p \text{ et } q\}$ et $(t, a, s) \in \delta$
   $\Leftrightarrow$ Par hypothèse, $t \in \eta(P, x)$ et $(t, a, s) \in \delta$
   $\Leftrightarrow$ Par définition de $\eta, s \in \eta(\eta(P, x), a) = \eta(P, xa)$

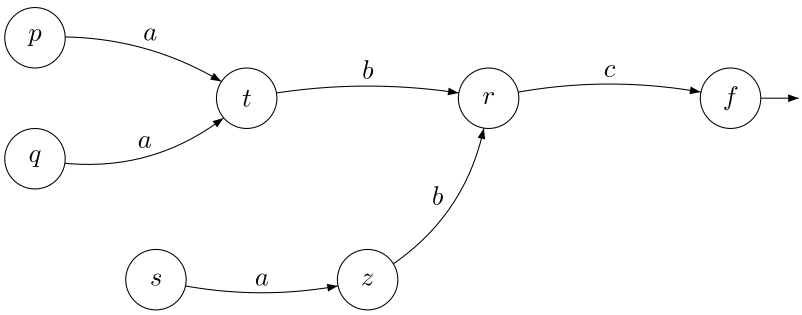
Ainsi, on a démontré que :

Automate quelconque $\Leftrightarrow$ Automate sans ε-transitions $\Leftrightarrow$ Automate déterministe. □

5. Minimisation d'un automate

5-a. Automate minimal

Exemple Soit l'automate :



On a alors $t \equiv z$ et $p \equiv q \equiv s$. $*$

***Définition* 13.** *Soit A un automate déterministe. Deux états p et q sont équivalents si et seulement si :*

$$p \equiv q \Leftrightarrow \forall x \in V^*, (\delta(p,x) \in F \Leftrightarrow \delta(q,x) \in F)$$

On note $[p]$ la classe d'équivalence de p.

Propriété 4:

$$p \equiv q \Longrightarrow (\forall x \in V^*,\ \delta(p,x) \equiv \delta(q,x))$$

Exemple Si on sait que $p = s$, alors $\delta(p,a) \equiv \delta(s,a)$. D'où :

$$t \equiv z$$

Théorème 12:
Soit A un automate déterministe. L'automate minimal équivalent à A est :

$$\mu(A) = \langle R, V, \eta, [i], G \rangle$$

$$\begin{array}{ccc} R & : & \text{Ensemble des classes d'équivalence de } \equiv \text{ sur } Q \\ G & : & \text{Ensemble des classes d'équivalence des états de } F \end{array}$$

$$\forall p \in Q,\ \forall a \in V,\ \eta([q],a) = [\delta(p,a)]$$

Démonstration Montrons que $L(A) = L(\mu(A))$.

$$\begin{array}{rcl} x \in L(A) & \Leftrightarrow & \delta(i,x) \in F \\ & \Leftrightarrow & [\delta(i,x)] \in G \\ & \Leftrightarrow & \eta([i],x) \in G \\ & \Leftrightarrow & x \in L(\mu(A)) \end{array}$$

$\mu(A)$ est minimal si A est initialement connecté, c'est à dire si tous ses états sont accessibles depuis l'état i. □

5-b. Construction de $\mu(A)$

Suite d'approximations de $\equiv$. $\equiv_k, k \geq 0$

$$p \equiv_k q \Leftrightarrow \forall x \in V^*, |x| \leq k$$

$$(\delta(p, x) \in F \Leftrightarrow \delta(q, x) \in F)$$

Alors

$$p \equiv_{k+1} q \Leftrightarrow [p \equiv_k q \ et \ \forall a \in V, \delta(p, a) \equiv_k \delta(q, a)]$$

Démonstration On démontre cette propriété :

– $\equiv_{k+1} q$ implique que :

1. $$p \equiv_k q$$

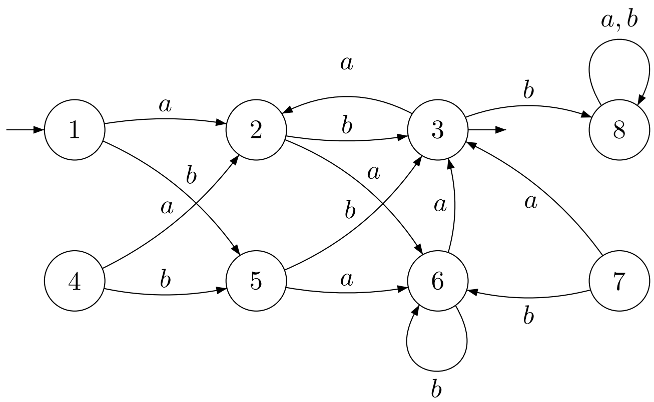
2. $$\begin{array}{rc} & \delta(p, x) \in F \Leftrightarrow \delta(q, x) \in F \ pour \ |x| \geq k + 1 \\ \Leftrightarrow & \delta(p, ay) \in F \Leftrightarrow \delta(q, ay) \in F, x = ay, |y| \leq k \\ \Leftrightarrow & \delta(\delta(p, a), y) \in F \Leftrightarrow \delta(\delta(q, a), y) \in F \\ \Leftrightarrow & \delta(p, a) \equiv_k \delta(q, a) \end{array}$$

–

$$\begin{array}{rc} \forall a \in V, & \delta(p, a) \equiv_k \delta(q, a) \\ \Leftrightarrow & \delta(\delta(p, a), x) \in F \Leftrightarrow \delta(\delta(q, a), x) \in F, |x| \leq k \\ \Leftrightarrow & \delta(p, ax) \in F \Leftrightarrow \delta(q, ax) \in F, |ax| \leq k + 1 \\ \Leftrightarrow & p \equiv_{k+1} q \end{array}$$

□

Exemple Soit l'automate :

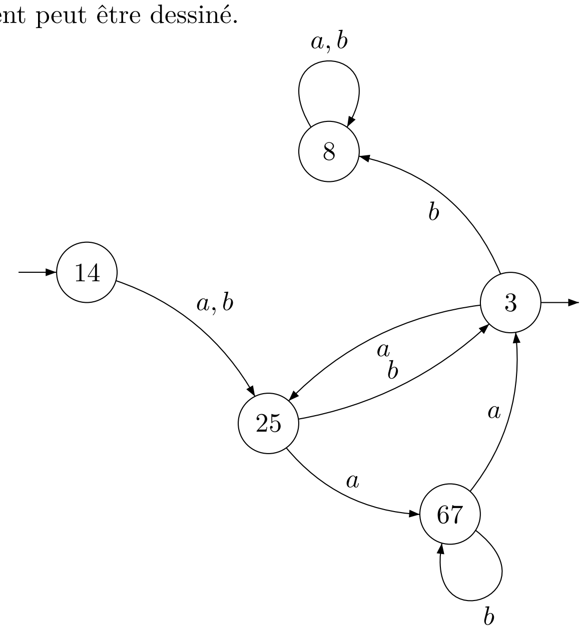


$$\equiv_0 : \ 1, 2, 4, 5, 6, 7, 8 \mid 3$$

$$\equiv_1 : \ 1, 4, 8 \mid 2, 5 \mid 6, 7 \mid 3$$

$$\equiv_2 : \ 1, 4 \mid 8 \mid 2, 5 \mid 6, 7 \mid 3$$

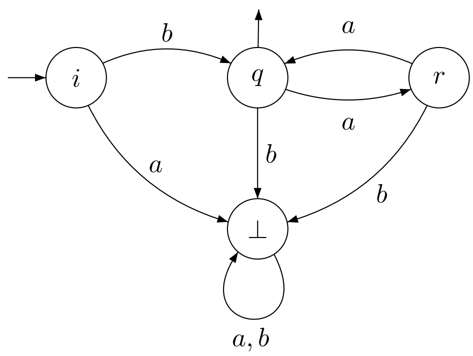
L'automate minimal équivalent peut être dessiné.



*

Exemple Intérêt en Algorithmique

Si on considère l'automate :



Grâce à cet automate, on peut déduire une écriture algorithmique. Voici la transposition de l'automate en écriture algorithmique :

1. Algorithme classique :

```
état := i
tant que mot pas fini faire
   C := caractère suivant;
   selon état faire
      cas i : si c ='b' alors
         état := q;
         sinon exit;
      cas q : si c ='a' alors
         état := r;
         sinon exit;
      cas r : si c='a' alors
         état := q;
         sinon exit;
   fin selon;
fin tant que;
reconnu := (état=q);
```

2. algorithme récursif :

```
reconnait_i :
                si mot fini renvoyer faux ;
                c := caractère suivant ;
                si c = 'b' renvoyer reconnait_q ;
                sinon renvoyer faux ;


reconnait_q :
                si mot fini renvoyer vrai ;
                c := caractère courant ;                          *
                si c := 'a' alors renvoyer reconnait_r ;
                sinon renvoyer faux ;


reconnait_r :
                si mot fini renvoyer faux ;
                c := caractère courant ;
                si c = 'a' renvoyer reconnait_q ;
                sinon renvoyer faux ;
```

6. *Clôture*

Théorème 13:
Si e_1 et e_2 sont deux langages réguliers :

1. $e = e_1 \cup e_2$
2. $e = e_1.e_2$
3. $e = e_1^*$
4. $e = e_1 \cap e_2$

sont des langages réguliers.

DÉMONSTRATION C'est immédiat par définition des expressions régulières. □

Théorème 14:
Si e_1 et e_2 sont deux langages hors contexte :

1. $e = e_1 \cup e_2$
2. $e = e_1.e_2$
3. $e = e_1^*$

sont des langages hors contexte. Mais $e = e_1 \cap e_2$ ne l'est pas forcément.

DÉMONSTRATION En effet, on peut donner un contre exemple :

$$\underbrace{\{a^k b^n c^n\}}_{Hors\ contexte} \cap \underbrace{\{a^p b^p c^l\}}_{Hors\ contexte} = \underbrace{\{a^n b^n c^n\}}_{pas\ Hors\ contexte}$$

7. Substitution

Définition 14. *Soient V et W deux vocabulaires. Une substitution associe à chaque lettre de V un langage sur W.*

$$\begin{array}{rcl} s : V & \longrightarrow & \mathcal{P}(W^*) \\ a & \longmapsto & L_a \subseteq W^* \end{array}$$

Exemple $V = \{a, b\}, W = \{0, 1\}$
$s(a) = \{01, 10\}$
$s(b) = \{000, 111\}$
On étend s aux langages sur V :

$$\begin{array}{rcl} s(\varepsilon) & = & \{\varepsilon\} \\ s(\omega) & = & a_1 a_2 \dots a_n \\ & = & s(a_1)s(a_2)\dots(a_n), a_i \in V \\ s(L) & = & \bigcup_{\omega \in L} s(\omega),\ L \subset V^* \\ s(ab) & = & s(a)s(b) \\ & = & \{01, 10\}.\{000, 111\} \\ & = & \{01000, 10000, 10111, 01111\} \\ s(b^*) & = & \bigcup_{\omega \in b^*} s(\omega) \\ & = & s\left(\bigcup_{n \geq 0} b^n\right) \\ & = & s(\varepsilon) + s(b) + s(bb) + \dots + s(b^n) + \dots \\ & = & (000 + 111)^* \\ & = & \{000, 111\}^* \end{array}$$

*

Définition 15. *La substitution s est dite régulière si et seulement si tous les $s(a)$ sont réguliers.*

Théorème 15:
Si s est régulière et L est régulier, alors :

$s(L)$ est régulier

Exemple Application du théorème :

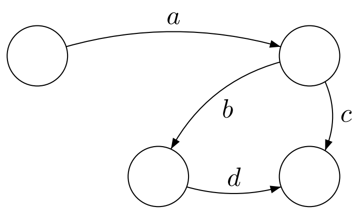
$$L = \{a^i b^j c^k / i + j = k \geq 0\}$$

On se demande si L est régulier. On peut procéder de plusieurs façons :

1. On applique le lemme de l'étoile :
$$z = a^p b^p c^{2p}$$
La seule décomposition possible est $u = a^i$ car $|uv| \leq p$: On voit que $uv^0w \notin L$. Ainsi, on peut dire que ce langage n'est pas régulier.
2. $$L \cap \underbrace{b^*c^*}_{\text{Régulier}} = \{\underbrace{b^n c^n}_{\text{Hors contexte}} / n \geq 0\}$$
Si L était régulier, alors $L \cap b^*c^*$ le serait aussi, donc L n'est pas régulier.
3. $s(a) = a, s(b) = a, s(c) = c$ régulières.
$$s(L) = \{a^n c^n / n \geq 0\} \text{ Hors contexte } \Rightarrow L \text{ non régulier}$$

Démonstration Preuve du Théorème
On peut utiliser plusieurs méthodes :

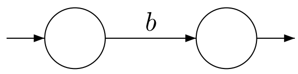
1. En utilisant les automates :



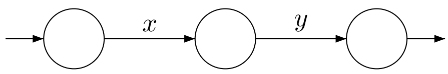
$$L(A) = L$$

$$\forall\ a, \exists A_a\ /L(A_a) = s(a)$$

$s(a)$:



$s(b)$:



Existe-t-il $A - s/L(A_s) = s(L)$?
A_s : je remplace chaque transition $\xrightarrow{a}$ pap A_a.

2. en utilisant l'expression régulière :

$$L = e$$

$$s(a) = l_a$$

$$\exists\ ?\ e_s = s(L)?$$

Idée de la démonstration par induction :

$$\begin{aligned}
s(L) &= \bigcup_{\omega \in L} s(\omega) \\
&= \sum_{\omega \in L} s(\omega) \\
&= \sum \prod s(a_i) \\
&= \sum \prod e_i \text{ qui est une expression régulière} \\
s(\omega) &= s(a_1)s(a_2)\ldots s(a_n) \\
&= \prod s(a_i)
\end{aligned}$$

3. en utilisant les grammaires :

$$G/L(G) = L$$

$$\forall a, G_a\ /\ L(G_a) = s(a)$$

$$\exists\ ?\ G_s\ /L(G_s) = s(L)?$$

Soit $G = \langle V, V_N, \mathcal{S}, R\rangle$ une grammaire qui engendre L.
Soit $\forall a \in V, G_a = \langle W, V_{N_a}, \mathcal{S}_a, R_a\rangle$ une grammaire qui engendre $s(a)$.
On suppose de plus que tous V_{N_a} et V_N sont disjoints, et que G et G_a sont linéaires à droite.

$$\begin{cases} X \longrightarrow \varepsilon \\ X \longrightarrow aY \\ x \longrightarrow Y \end{cases}$$

On construit

$$G_s = \langle W, V_{N_s}, \mathcal{S}, R_s \rangle$$

$$V_{N_s} = V_N \cup \left(\bigcup_{a \in V} V_{N_a} \right)$$

$$\begin{aligned} R_s = \quad & \{A \longrightarrow \mathcal{S}aB / A \longrightarrow aB \in R\} \\ \cup \quad & \{A \longrightarrow \varepsilon / A \longrightarrow \varepsilon \in R\} \\ \cup \quad & \{A \longrightarrow B / A \longrightarrow B \in R\} \\ \cup \quad & \bigcup_{a \in V} R_a \end{aligned}$$

□

Exemple $L = a^*b$.

$$\begin{cases} \mathcal{S} \longrightarrow a\mathcal{S} | b\mathcal{T} \\ \mathcal{T} \longrightarrow \varepsilon \end{cases}$$

$G_s : \begin{cases} \mathcal{S} \longrightarrow \mathcal{S}a\mathcal{S} | \mathcal{S}b\mathcal{T} \\ T \longrightarrow \varepsilon \end{cases}$

$s(a) = (x+y).y$:

$$\begin{cases} \mathcal{S}_a \longrightarrow x\mathcal{T}_a | y\mathcal{T}_a \\ \mathcal{T}_a \longrightarrow y\mathcal{U}_a \\ \mathcal{U}_a \longrightarrow \varepsilon \end{cases}$$

$s(b) = xy^*z$:

$$\begin{cases} \mathcal{S}_b \longrightarrow x\mathcal{T}_b \\ \mathcal{T}_b \longrightarrow y\mathcal{T}_b | U_b \\ \mathcal{U}_b \longrightarrow z\mathcal{V}_b \\ \mathcal{V}_b \longrightarrow \varepsilon \end{cases}$$

On a bien G_s qui engendre $s(L)$, mais les grammaires de la sorte :

$$\mathcal{S} \longrightarrow \mathcal{S}a\mathcal{S}$$

sont non linéaires à droite ! On va essayer de trouver un procédé algorithmique pour les transformer en grammaires linéaires à droite. Soient :

$$\begin{array}{rcl} \mathcal{S} & \longrightarrow & [\mathcal{S}_a\mathcal{S}] \mid [\mathcal{S}_b\mathcal{T}] \\ \mathcal{T} & \longrightarrow & \varepsilon \\ [\mathcal{S}_a\mathcal{S}] & \longrightarrow & x[\mathcal{T}_a\mathcal{S}] \mid y[\mathcal{T}_a\mathcal{S}] \\ [\mathcal{T}_a\mathcal{S}] & \longrightarrow & y[\mathcal{U}_a\mathcal{S}] \\ [\mathcal{U}_a\mathcal{S}] & \longrightarrow & \mathcal{S} \\ [\mathcal{S}_b\mathcal{T}] & \longrightarrow & x[\mathcal{T}_b\mathcal{T}] \\ [\mathcal{T}_b\mathcal{T}] & \longrightarrow & y[\mathcal{T}_b\mathcal{T}] \mid [\mathcal{U}_b\mathcal{T}] \\ [\mathcal{U}_b\mathcal{T}] & \longrightarrow & z[\mathcal{U}_b\mathcal{T}] \\ [\mathcal{V}_b\mathcal{T}] & \longrightarrow & \mathcal{T} \end{array}$$

Il reste à finir la démonstration . . . ∗

8. Lien avec le cours d'architecture

8-a. Automate de Mealy

***Définition* 16.** DÉFINITION FORMELLE
Un automate de Mealy est un automate de la forme :

$$\langle Q, V, W, i, \delta, s \rangle$$

Avec :

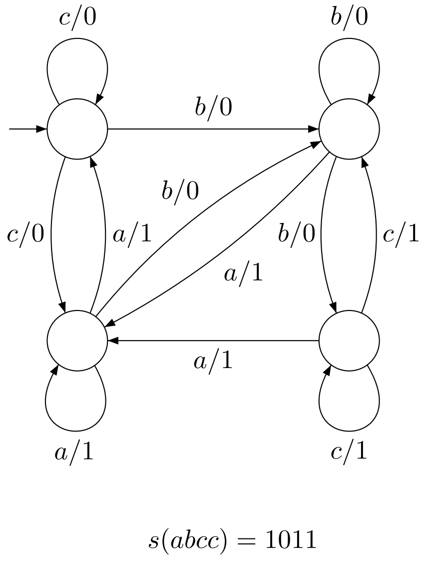
$$\begin{array}{lcl} Q & : & \textit{états} \\ V & : & \textit{alphabet d'entrée} \\ W & : & \textit{alphabet de sortie} \\ i & : & \textit{état initial} \\ \delta & : & \textit{fonction de transition} \\ s & : & \textit{fonction de sortie} \end{array}$$

Remarque Un automate de Mealy est déterministe *

$$\begin{array}{rcl} \delta : Q \times V & \longrightarrow & Q \\ s : Q \times V & \longrightarrow & W \end{array}$$

Dans les automates de Mealy, il n'y a qu'une seule sortie par transition.

Exemple Soit l'automate de Mealy suivant :



$$s(abcc) = 1011$$

8-b. Automate de Moore

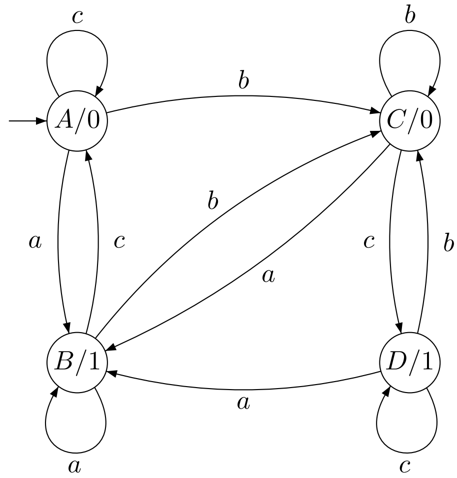
On définit un automate de Moore par :

$$\langle Q, V, W, i, \delta, s \rangle$$

C'est la même chose que l'automate de Mealy sauf que :

$$s : Q \longrightarrow W$$

Exemple On dessine un automate de Moore :

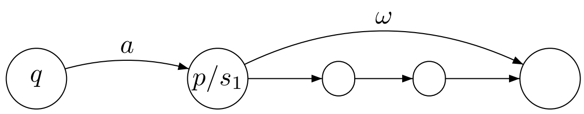


Cet automate est une machine de Moore. Il n'y a qu'une seule sortie par état. L'automate est complet et déterministe et ne possède pas d'état final. Une machine de Moore calcule une fonction :

$$s(abcc) = 1011$$

Exemple $s(A) = 0$.
On peut faire une extension aux mots :

$$\begin{array}{rcl} s(q, \varepsilon) & = & \varepsilon \\ s(q, a\omega) & = & s(\delta(q,a)).s(\delta(q,a),\omega) \end{array}$$



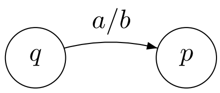
*

8-c. De Moore à Mealy

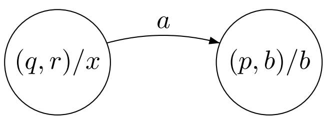
Les deux modèles sont équivalents.
On part d'une machine de Mealy Me et on veut construire une machine de Moore qui fait la même chose :

$$Q' = \underset{\{(q,r)\}}{Q_e} \times W$$

Machine de Mealy Me :



Machine de Moore $M' \Leftrightarrow \forall r \in W$

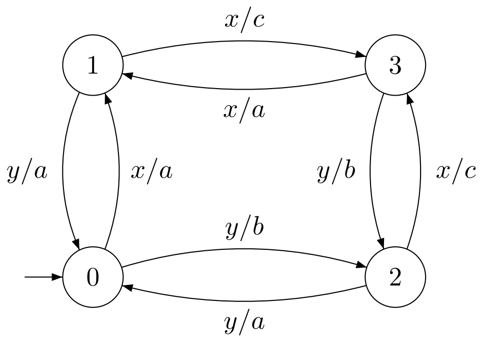


$$\delta'((q,r),a) = (\delta(q,a), s(q,a))$$
$$s'((q,r)) = r$$

8-d. Minimisation d'une machine de Mealy

Il faut trouver les états q et p tels que $q \equiv p$ pour les machines de Mealy.
$\forall x \in V^*$, on veut la même réponse pour p et q :

$$s(q, x) = s(p, x)$$



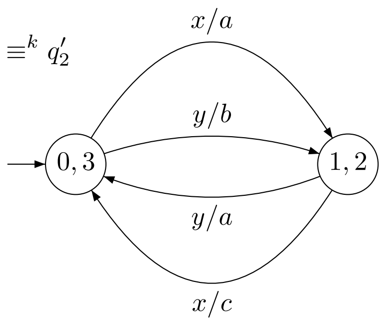
$$xyyx = aabc$$

$$\begin{array}{lll} q_1 & \equiv_{k+1} & q_2 \\ & \Leftrightarrow & \begin{cases} q_1 \equiv_k q_2 \\ q_1 \xrightarrow{a/S} q'_A \end{cases} \\ & \Rightarrow & \exists\ q'_2 \ / \ q_2 \xrightarrow{a/S} q'_2 \ et \ q'_1 \equiv^k q'_2 \end{array}$$

$$q_1 \equiv_0 \{0, 1, 2, 3\}$$

0 n'est pas équivalent à 2 car on n'a pas la même sortie.

$$\begin{array}{lll} q_1 & \equiv_1 & \{0, 3\}\{1, 2\} \\ & \equiv_2 & \{0, 3\}\{1, 2\} \end{array}$$



8-e. Exemple d'automate appliqué à l'imagerie