Mathématiques 2 : Langages Formels

HE Vinci : Informatique de gestion

7 mai 2023

Alphabet, Mots et Langages : Introduction

Principale application des langages formels :

$\rightarrow$ Les langages de programmations

- Généralement un langage doit être compilé pour pouvoir être exécuté
- Quelle(s) condition(s) afin d'assurer la faisabilité d'un compilateur pour un langage ?

$\rightarrow$ le langage doit être **regulier** (formel)

Alphabets

Un **alphabet** est un **ensemble fini non vide** souvent noté Σ

Les éléments d'un alphabet peuvent être de n'importe quelle nature :

- des caractères : $\{a, b, c\}$
- des mots-clefs Java : $\{\texttt{if}, \texttt{then}, \texttt{else}, \texttt{while}, \texttt{return}, \texttt{for}\}$
- des chiffres : $\{3, 5, 2\}$
- $\cdots$

Par commodité, nous appellerons les éléments de l'alphabet

« **lettres** », ou « **symboles terminaux** »

.

Mots

Un **mot** est une **suite finie** (éventuellement vide) **de lettres**.

- En général, un mot s'écrit par simple juxtaposition de ses lettres.
  Exemples de mots sur $\Sigma = \{a, b, c\}$:
  - *abbcabcbb*
  - *aa*
  - *abc*

- La **longueur** d'un mot *m* est le nombre de lettres qu'il contient.
  - Elle sera désignée par $|m|$.
  - Exemple : Si $\Sigma = \{a, b, c\}$, alors $|abbcabcbb| = 9$.

- Le **mot vide** sera noté ε. C'est le seul mot de longueur 0.

Le symbole ε n'appartient pas à l'alphabet.

Mots

Un **mot** est une **suite finie** (éventuellement vide) **de lettres**.

- Si m et p sont des mots, alors la concaténation de m et p est mp.
  Propriété : Le mot vide ε est neutre pour la concaténation :
  $$\forall p \left(p\varepsilon = p = \varepsilon p \right)$$
- On écrira m^k pour $m \cdots m$ (k fois).

Exemples :

Si on a les deux mots $m = acba$ et $p = abc$ définit sur l'alphabet $\Sigma = \{a, b, c\}$, alors :

1. $mp = acbaabc$
2. $p^2 = abcabc$
3. $p^0 = \varepsilon$
4. $p\varepsilon = abc = \varepsilon p$
5. $pm^2p = abcacbaacbaabc$

Langages : définition et exemple

On appelle **langage** sur un alphabet Σ **tout ensemble de mots** sur Σ.

Exemple :

Sur l'alphabet $\Sigma = \{a, b, c\}$, on peut définir les langages

1. $L_1 = \{abc,\ bca,\ a,\ aaa\}$
2. $L_2 = \{aa,\ bca,\ abc,\ (bbc)^3,\ \varepsilon\}$
3. $L_3 =$ ensemble des mots contenant au moins deux a
4. $\{\varepsilon\}$: langage particulier sur $\Sigma \rightarrow$ il contient 1 mot.
5. $\varnothing$: plus petit langage sur $\Sigma \rightarrow$ il ne contient aucun mot.
6. $\cdots$

Il existe une infinité de langage sur un alphabet Σ.

Remarque :

Il est commode de confondre les lettres et les mots de longueur 1.

$\rightarrow$ l'alphabet Σ peut être vu comme le langage formé de tous les mots de longueur 1.

Opérations sur les langages

Opérations construites à partir d'opérations ensemblistes

Soit deux langages L_1 et L_2 sur un alphabet Σ, alors

1. $L_1 \cup L_2 =$ langage des mots appartenant à L_1 **OU** à L_2 (ou aux deux)
2. $L_1 \cap L_2 =$ langage des mots appartenant à L_1 **ET** à L_2
3. $L_1 - L_2 =$ langage des mots appartenant à L_1 **mais pas à** L_2

Exemple :

Soit l'alphabet $\Sigma = \{a, b, c\}$ et les langages sur Σ

- $L_1 = \{abc,\ bca,\ a,\ aaa\}$
- $L_2 = \{aa,\ bca,\ abc,\ (bbc)^3,\ \varepsilon\}$,

Alors

1) $L_1 \cup L_2 = \{abc,\ bca,\ a,\ aaa,\ aa,\ (bbc)^3, \varepsilon\}$
2) $L_1 \cap L_2 = \{abc,\ bca\}$
3) $L_1 - L_2 = \{a,\ aaa\}$.

Opérateur de concaténation : •

Soit deux langages L_1 et L_2 sur un alphabet Σ, alors

$$L_1 \bullet L_2 \quad = \quad \text{ensemble des mots obtenus en concaténant un mot de } L_1 \text{ avec un mot de } L_2 \text{ en commençant par le mot de } L_1$$

Exemple :

Soit l'alphabet $\Sigma = \{a, b, c\}$ et les langages sur Σ

- $L_1 = \{abc,\ bca,\ a,\ aaa\}$
- $L_2 = \{aa,\ bca,\ abc,\ (bbc)^3,\ \varepsilon\}$,

Alors

$$\begin{aligned} L_1 \bullet L_2 \quad = \quad & \{abcaa, abcbca, abcabc, abc(bbc)^3, abc\varepsilon = abc,\\ & bcaaa, bcabca, bcaabc, bca(bbc)^3, bca\varepsilon = bca,\\ & aaa, abca, aabc, a(bbc)^3, a\varepsilon = a,\\ & aaaaa, aaabca, aaaabc, aaa(bbc)^3, aaa\varepsilon = aaa\} \end{aligned}$$

$\rightarrow$ chaque mot de L_1 concaténé avec chaque mot de L_2.

Propriété : Le langage $\{\varepsilon\}$ est neutre pour l'opération •.

Opérateur de concaténation : Notation exponentielle

Si L est un langage sur un alphabet Σ, alors

- on notera L^2 le langage $L \bullet L$,
- on notera L^3 le langage $L^2 \bullet L = L \bullet L \bullet L$,
- $\cdots$
- On aura naturellement que $L^0 = \{\varepsilon\}$

Exemple :

Soit le langage $L = \{ab,\ a,\ abc,\ bc\}$ sur l'alphabet $\Sigma = \{a, b, c\}$, alors

1. Le langage L^2 est

$$\begin{aligned} L^2 = L \bullet L \quad = \quad & \{\boldsymbol{ab}ab,\ \boldsymbol{ab}a,\ \boldsymbol{ab}abc,\ \boldsymbol{ab}bc, \\ & \boldsymbol{a}ab,\ \boldsymbol{a}a,\ \boldsymbol{a}abc,\ \boldsymbol{a}bc, \\ & \boldsymbol{abc}ab,\ ,\boldsymbol{abc}a,\ \boldsymbol{abc}abc,\ \boldsymbol{abc}bc, \\ & \boldsymbol{bc}ab,\ \boldsymbol{bc}a,\ \boldsymbol{bc}abc,\ \boldsymbol{bc}bc\} \end{aligned}$$

$\rightarrow$ **chaque mot** de L concaténé avec chaque mot de L.

Opérateur de concaténation : Notation exponentielle

Soit le langage $L = \{ab,\ a,\ abc,\ bc\}$ sur l'alphabet $\Sigma = \{a, b, c\}$, alors

2. $abca \in L^2 \cap L^3$ car $\begin{cases} abca \in L^2 \text{ car } abca \text{ concaténation de } abc \text{ avec } a \\ abca \in L^3 \text{ car } abca \text{ concaténation de } a,\ bc \text{ et } a \end{cases}$

3. $abaabab \in L^4$ car $abaabab$ concaténation de ab, a, ab et ab.

Opérateurs $+$ et $*$

Si L est un langage sur un alphabet Σ, on note

L^+ : le langage constitué de tous les mots obtenus en concaténant un nombre fini non nul de mots de L.

$$\boxed{L^+ = L \cup L^2 \cup L^3 \cup L^4 \cup \cdots}$$

L^* : le langage constitué de tous les mots obtenus en concaténant un nombre fini quelconque de mots de L

$$\boxed{L^* = \boldsymbol{L^0} \cup L \cup L^2 \cup L^3 \cup L^4 \cup \cdots = \{\varepsilon\} \cup L^+}$$

Propriétés :

- **Les langages L^+ et L^* sont égaux ssi le mot ε appartient à L**
- Le langage Σ^* est le langage constitué de tous les mots sur Σ.

Exemple :

Soit le langage $L = \{a\}$ sur l'alphabet $\Sigma = \{a, b, c\}$, alors

- L^+ est le langage des mots formés d'un nombre quelconque non nul de a.
- $\left(L^2\right)^*$ est le langage des mots formés d'un nombre pair de a.

Expressions Régulières : Introduction

Pour décrire un langage régulier avec une infinité de mots $\rightarrow$ 3 outils :

1. Les « **expressions régulières** » :
   - outil permettant de décrire, de manière compacte, certains types de langages.
   - définissent, via une chaîne de caractères, le « modèle »des mots appartenant au langage.
2. Des « **systèmes générateurs** » ou « **grammaires** » :
   - ensemble de règles formelles décrivant la ou les manière de construire les mots du langage.
3. Des « **reconnaisseurs** » ou « **automates** » :
   - schémas d'algorithmes permettant de décider si oui ou non un mot proposé appartient au langage.

Expressions Régulières : Définition (par induction)

Pour définir ce qu'est une expression régulière, nous avons besoin

- De l'ensemble Δ contenant les 6 symboles : $\varnothing$, λ, $($, $)$, $*$, $|$
- D'un alphabet Σ disjoint de Δ (donc aucun symbole de Δ n'est une lettre)

Alors une **expression régulière** (ou « **rationnelle** ») sur l'alphabet Σ est un mot du langage $ER(\Sigma)$ défini par induction par

1) Les expressions primitives (expressions régulières de longueur 1) :
   - $\varnothing$
   - λ
   - $\boldsymbol{x}$ pour $x \in \Sigma$ (n'importe quelle lettre de Σ est une expression primitive)
2) Les expressions composées (construites à l'aide des symboles de Δ à partir d'expressions primitives ou d'autres expressions composées) :
   - Si α et β sont des expressions régulières quelconques alors
     - $\alpha\beta$ est une expression régulière.
     - $(\alpha|\beta)$ est une expression régulière.
     - $(\alpha)^*$ est une expression régulière
   - Si α est une expression régulière primitive ou parenthésée (entre parenthèses) alors α^* est une expression régulière.

Expressions Régulières : Exemples

Soit l'alphabet $\Sigma = \{a, b, c\}$

1) Expressions régulières primitives sur Σ :

- Les symboles $\varnothing$ et λ
- Les lettres de l'alphabet Σ : $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$

2) Expressions régulières composées sur Σ

- $(a|b) \longrightarrow a$ et b sont des expressions régulières primitives donc $(a|b)$ aussi
- $(a|b)\,c \longrightarrow (a|b)$ et c sont des expressions régulières donc $(a|b)\,c$ aussi
- $(a|b)\,c^*$
  $\longrightarrow$ $(a|b)$ et c^* sont des expressions régulières (c : expression primitive)
  $\longrightarrow$ $(a|b)\,c^*$ est une expression régulière composée
- $(a|b)^*(ccba)^*\lambda$
  $\longrightarrow$ $(a|b)$, a b, c et λ sont des expressions primitives
  $\longrightarrow$ $(a|b)^*$ et $(ccba)^*$ sont des expressions régulières composées
  $\longrightarrow$ $(a|b)^*(ccba)^*\lambda$ est une expression régulière
- $a^*\left((a|b)^*ccba^*\middle|\lambda\right)bc^*b(bc^*)^*$
- $\ldots$

Expressions Régulières : Contre-Exemples

- (aa) :
  - $\longrightarrow$ Les parenthèses ne sont utilisées que dans deux cas : $(\alpha|\beta)$ ou $(\alpha)^*$.
  - $\longrightarrow$ Cette expression ne peut être apparentée à aucun de ces 2 cas.

- $(aa|*)$:
  - $\longrightarrow$ $(\alpha|\beta)$ est une expression régulière ssi α et β sont des expressions régulières
  - $\longrightarrow$ le symbole $*$ n'est pas une expression régulière correcte.

- $(aa|\ |aa)$:
  - $\longrightarrow$ $(\alpha|\beta)$ expression régulière ssi α et β soit expressions régulières
  - $\longrightarrow$ l'expression $|aa$ n'est pas une expression régulière correcte.

- ${}^*(aa|ab^*a)$:
  - $\longrightarrow$ $*$ doit être précédé soit d'une expression régulière soit primitive soit parenthésée.
  - $\longrightarrow$ le $*$ au début de l'expression n'est précédé par rien.

Expressions régulières : Langage associé

A toute expression régulière sur un alphabet Σ correspond un langage sur Σ, appelé **langage associé**.

Si α est une expression régulière, alors le langage associé à α est noté $\boldsymbol{L_\alpha}$

Ce langage associé est déterminé de manière inductive comme suit :

Expression	**Langage associé**	
$\varnothing$	$\{\}$	(Le langage vide)
λ	$\{\varepsilon\}$	(Le langage ne contenant que le mot vide)
x $(x \in \Sigma)$	$\{x\}$	(Le langage ne contenant que le mot x)
$\alpha\beta$	$L_\alpha \bullet L_\beta$	(Le langage des mots construit en concaténant un mot défini par α avec un mot défini par β)
$(\alpha\|\beta)$	$L_\alpha \cup L_\beta$	(Tous les mots définis soit par α soit par β)
$(\alpha)^*$ α^*	L_α^*	(Le langage des mots obtenus par concaténation d'un nombre quelconque de mots définis par α)

Expressions Régulières : Langages Equivalents

Des expressions régulières sont dites **équivalentes** si elles ont le **même langage associé**.

Exemple :

Sur l'alphabet Σ les expressions régulières $\alpha = a(ba|a)$ et $\beta = (ab|a)a$ sont équivalentes.
En effet les langages associés à ces deux langages sont

- $L_\alpha = \{\boldsymbol{a}ba,\ \boldsymbol{a}a\} = \{aba,\ aa\}$
- $L_\beta = \{\boldsymbol{ab}a,\ \boldsymbol{a}a\} = \{aba,\ aa\} = L_\alpha$

Ces deux langages sont les mêmes $\longrightarrow$ α et β sont équivalentes.

Expressions Régulières : Langages Equivalents

Abus fréquents (et commodes) de notations :

1. Les expressions régulières $\left(\alpha \middle| (\beta|\gamma)\right)$ et $\left((\alpha|\beta) \middle| \gamma\right)$ sont équivalentes quelles que soient α, β, γ
   $\longrightarrow$ on va les écrire de manière raccourcie par $\left(\alpha \middle| \beta \middle| \gamma\right)$
2. On écrira $(\alpha)^+$ pour raccourcir les expressions $\alpha\,(\alpha)^*$ et $(\alpha)^*\,\alpha$
3. On écrira $(\alpha)^k$ pour raccourcir $\underbrace{\alpha\cdots\alpha}_{k \text{ fois}}$

Expressions régulières : Langages Réguliers

Expressions régulières

$\longrightarrow$ premier moyen de déterminer quand un langage est compilable (régulier)

Un langage sur un alphabet Σ est dit **régulier** (ou **rationnel**) si

$$\boxed{\text{il est le langage associé à une expression régulière sur } \Sigma}$$

Exemples :

1. Sur l'alphabet $\Sigma = \{a,\ b,\ c\}$, le langage constitué de tous les mots qui commencent et finissent par 'a' est régulier :

   Un mot sur Σ commencera et finira par 'a'

   soit si c'est le mot a

   soit si c'est un mot qui commence par a suivit par un nombre quelconque de lettres de Σ et qui se termine par a autrement dit un mot définit par l'expression régulière $a(a|b|c)^*a$

   $\longrightarrow$ langage régulier car associé à l'expression régulière $\left(a \,\middle|\, a(a|b|c)^*a\right)$

Expressions régulières : Langages Réguliers

Exemple :

2. La vie d'un document dans une bibliothèque peut être décrite par l'expression régulière :

$$\textit{acquérir}\,(\textit{sortir rentrer})^*(\textit{sortir}|\textit{vendre}|\textit{archiver})$$

En effet, un livre va

1) être acquis par la bibliothèque
2) être emprunté un nombre quelconque de fois (il sera loué puis rendu).
3) terminer sa vie à la bibliothèque trois façons possibles :

   soit un locataire ne rendra pas le livre (il n'y aura pas de rentrée après la sortie)
   soit la bibliothèque va le vendre
   soit la bibliothèque va l'archiver.

Expressions Régulières : Examen de Juin 2017

Soit les langages $L_1,\ L_2$ et L_3 construits sur l'alphabet $\Sigma = \{a, b, c, d\}$.

- $L_1 = aa^*(c|d)^*(cd)^*$
- $L_2 = (aa)^*(c|d)^*(cd)^*$
- $L_3 = \left\{(aa)^n(c|d)^p(cd)^n \,\middle|\, n, p \in \mathbb{N}\right\}$

Montrez que les langages $L_1,\ L_2$ et L_3 sont deux à deux différents.

Solution :

On a $L_1 \neq L_2$ si L_1 contient un mot qui n'est pas dans L_2. Or

1. Le mot a est dans L_1 car $a = aa^0(c|d)^0(cd)^0$
2. Le mot a n'est pas dans L_2 ni dans L_3 car les mots de ces langages ne peuvent contenir que des groupes aa et auront donc un nombre pair de a.
3. Le mot aa est dans L_2 car $aa = (aa)^1(c|d)^0(cd)^0$
4. Le mot aa n'est pas dans L_3 car pour obtenir seulement aa, il faudrait prendre $n = 1$ et $p = 0$ mais dans ce cas le mot obtenu est $aacd \neq aa$.

Donc par 1. et 2., on a $L_1 \neq L_2$ et $L_1 \neq L_3$.Et par 3. et 4. on a $L_2 \neq L_3$
Conclusion les langages $L_1,\ L_2$ et L_3 sont deux à deux différents.

Grammaires Régulières : Introduction

- But de ce chapitre : déterminer si un langage est régulier (compilable).
- Premier outil vu dans la section précédent : les expressions régulières.
- Dans cette section : deuxième outil : les grammaires régulières.
- Grammaire régulière : ensemble de règles formelles décrivant la ou les manières de construire les mots d'un langage.

Grammaires Régulières : Définition

Une **grammaire régulière** est composée par :

1. Un alphabet Σ (appelé ensemble des **symboles terminaux**)
2. Un ensemble fini N disjoint de Σ dont les éléments sont les **symboles non terminaux**.
   Ceux-ci sont généralement noté $< \text{lettre majuscule} >$:
   Par exemple $< A >$.
3. $< S >$ un élément de N appelé **axiome** ou **point de départ** des "dérivations"
4. Un ensemble fini P dont les éléments sont les règles de dérivation.
   Une règle de dérivation peut-être de la forme
   - $< A > \rightarrow m$ où $< A > \in \mathrm{N}$ et $m \in \Sigma^*$
   - $< A > \rightarrow m < B >$ où $< A >, < B > \in \mathrm{N}$ et $m \in \Sigma^*$
   - $< A > \rightarrow < B >$ où $< A >, < B > \in \mathrm{N}$

   où la flèche $\rightarrow$ signifie "peut être remplacer par".

Une **grammaire régulière** est donc un quadruplet $G = (\Sigma, \mathrm{N}, < S >, \mathrm{P})$

Grammaires Régulières : Langage engendré

A partir d'un grammaire on peut engendrer un langage :

Le langage L « engendré » par la grammaire $\boldsymbol{G}$ est celui des mots auxquels on peut arriver en appliquant un nombre fini de « dérivations » à partir de l'axiome.

Grammaires Régulières : Exemple

Soit l'alphabet $\Sigma = \{a,\ b,\ c\}$ et la grammaire dont les règles de dérivation sont

$$\boxed{\begin{array}{lcl} <S> & \rightarrow & a<A> \mid b<B> \\ <A> & \rightarrow & cc<A> \mid b<B> \mid \varepsilon \\ <B> & \rightarrow & ab<B> \mid b<S> \mid c \end{array}}$$

Alors

- ensemble des symboles "non terminaux" : $\mathrm{N} = \{<S>,\ <A>,\ <B>\}$
- $<S>$ est l'axiome, le point de départ des "dérivations"
- l'ensemble P des règles de dérivation possède 8 éléments.

  En effet $<S>\rightarrow a<A> \Big| b<B>$ est un raccourci pour

  $<S>\rightarrow a<A>$

  $<S>\rightarrow b<B>$

  De plus, la flèche $\rightarrow$ signifie "peut être remplacer par".

  Donc, par exemple, on peut "remplacer" $<S>$ par $a<A>$

On commencera toujours une suite de dérivation par l'axiome.

Grammaires Régulières : Exemple

Exemple de dérivations possibles à partir de cette grammaire :

1) Dérivation 1 :

Grammaire

$$
\begin{array}{lcl}
<S> & \rightarrow & a<A> \mid b<B> \\
<A> & \rightarrow & cc<A> \mid b<B> \mid \varepsilon \\
<B> & \rightarrow & ab<B> \mid b<S> \mid c
\end{array}
$$

Dérivation

$$
\begin{array}{lcl}
<S> & \rightarrow & a<A> \\
 & \rightarrow & acc<A> \\
 & \rightarrow & acc\varepsilon
\end{array}
$$

$\longrightarrow$ **le mot *acc* appartient au langage engendré**

Grammaires Régulières : Exemple

2) Dérivation 2 :

Grammaire

$$\begin{array}{lcl} \langle S \rangle & \rightarrow & a \langle A \rangle \mid b \langle B \rangle \\ \langle A \rangle & \rightarrow & cc \langle A \rangle \mid b \langle B \rangle \mid \varepsilon \\ \langle B \rangle & \rightarrow & ab \langle B \rangle \mid b \langle S \rangle \mid c \end{array}$$

Dérivation

$$\begin{array}{lcl} \langle S \rangle & \rightarrow & a \langle A \rangle \\ & \rightarrow & acc \langle A \rangle \\ & \rightarrow & accb \langle B \rangle \\ & \rightarrow & accbc \end{array}$$

$\longrightarrow$ **le mot *accbc* appartient au langage engendré**

Grammaires Régulières : Exemple

3) Dérivation 3 :

Grammaire

$< S >$	$\rightarrow$	$a < A >$	$b < B >$	
$< A >$	$\rightarrow$	$cc < A >$	$b < B >$	ε
$< B >$	$\rightarrow$	$ab < B >$	$b < S >$	c

Dérivation

$$
\begin{array}{lcl}
< S > & \rightarrow & b < B > \\
 & \rightarrow & bb < S > \\
 & \rightarrow & bba < A > \\
 & \rightarrow & bbacc < A > \\
 & \rightarrow & bbaccb < B > \\
 & \rightarrow & bbaccbab < B > \\
 & \rightarrow & bbaccbabc
\end{array}
$$

$\longrightarrow$ **le mot** ***bbaccbabc*** **appartient au langage engendré**

Conclusion :

Une suite de dérivation

- commence par l'axiome
- aboutit à un mot lorsqu'il n'y a plus de symbole non terminal après application d'une règle de dérivation.

Grammaire normalisée

Une grammaire régulière est dite **normalisée** si tous les membres droits de ses règles de dérivation sont d'une des formes suivantes :

- ε
- x où $x \in \Sigma$ (x est une lettre de l'alphabet Σ)
- $x < Z >$ où $x \in \Sigma$ et $< Z >$ est un symbole non terminal ($< Z > \in \mathrm{N}$)

Toute grammaire régulière peut être normalisée

Une règle non normale de la forme $< X > \rightarrow ab < Z >$ peut être remplacée par les règles (normales) :

$$< X > \rightarrow a < Y >$$
$$< Y > \rightarrow b < Z >$$

$\longrightarrow$ Il a fallu ajouter le symbole non terminal $< Y >$ à l'ensemble P des symboles non terminaux

Conclusion :

Les grammaires régulières normalisées engendrent donc les mêmes langages que les grammaires régulières.

Grammaire et expression régulière

On a propriété suivante

Tout langage régulier peut être engendré par une grammaire régulière

Pour montrer cela,

$\longrightarrow$ on va passer par les expressions régulières.

$\longrightarrow$ un langage est régulier s'il peut être associé à une expression régulière.

$\longrightarrow$ si pour chaque règle de construction d'une expression régulière je peux trouver une règle de dérivation équivalente pour les grammaires, on aura que la grammaire ainsi crée engendrera le langage régulier associé à cette expression régulière !

Dans la suite on va considérer que deux grammaires différentes

$$G_\alpha : \boxed{\begin{array}{ccc} <S_\alpha> & \rightarrow & \cdots \\ \cdots & \cdots & \cdots \end{array}} \text{ et } G_\beta : \boxed{\begin{array}{ccc} <S_\beta> & \rightarrow & \cdots \\ \cdots & \cdots & \cdots \end{array}}$$

ont des symboles non terminaux différents (aucun symbole non terminal commun)

Grammaire et expression régulière

1) Pour l'expression régulière primitive $\varnothing$: $< S > \to\ x < S >$ où x est une lettre quelconque de Σ.
   $\longrightarrow$ règle non terminale
   $\longrightarrow$ une grammaire avec cette unique règle ne génèrera aucun mot.
   $\longrightarrow$ génère le langage vide comme $\varnothing$.

2) Pour l'expression régulière primitive λ : $< S > \to\ \varepsilon$
   $\longrightarrow$ une grammaire avec cette unique règle ne peut générer que le mot vide ε
   $\longrightarrow$ génère le langage ne contenant que le mot vide comme λ

3) Pour l'expression régulière primitive x $(x \in \Sigma)$: $< S > \to x$
   $\longrightarrow$ une grammaire avec cette unique règle ne peut générer que le mot x
   $\longrightarrow$ génère le langage $\{x\}$ comme x

Grammaire et expression régulière

4) Pour l'expression composée $(\alpha|\beta)$: à partir des grammaires G_α et G_β on va
   1. Ajouter un nouveau symbole non terminal $< S >$ comme axiome
   2. Ajouter la règle de dérivation $< S > \rightarrow < S_\alpha > \Big| < S_\beta >$
   3. Garder toutes les règles de dérivation associées aux grammaires G_α et G_β.

   Donc

   $\longrightarrow$ à partir de l'axiome on pourra soit partir vers la grammaire générée par α soit vers la grammaire générée par β.

   $\longrightarrow$ on pourra générer soit un mot associé à α soit un mot associé à β.

   $\longrightarrow$ on générera le même langage que $(\alpha|\beta)$

Grammaire et expression régulière

5) Pour l'expression composée $\alpha\beta$: à partir des grammaires G_α et G_β on va
   1. Garder $< S_\alpha >$ comme axiome
   2. Remplacer toute règle terminale $< X > \rightarrow u$ de G_α par la règle $< X > \rightarrow u < S_\beta >$
   3. Garder toutes les autres règles de dérivation de G_α et toutes les règles de dérivation de G_β.

   Donc

   $\longrightarrow$ on partira de l'axiome de G_α

   $\longrightarrow$ on génèrera un mot associé à α

   $\longrightarrow$ on repartira de l'axiome de G_β et on génèrera un mot associé à β.

   $\longrightarrow$ on pourra généré un mot qui est la concaténation d'un mot associé à α avec un mot associé à β.

   $\longrightarrow$ on générera le même langage que $\alpha\beta$

Grammaire et expression régulière

6) Pour l'expression composée $(\alpha)^*$: à partir de la grammaires G_α on va
    1. Ajouter un nouveau symbole non terminal $< S >$ comme axiome
    2. Ajouter la règle de dérivation $< S > \rightarrow \varepsilon \Big| < S_\alpha >$
    3. Remplacer toute règle terminale $< X > \rightarrow u$ de G_α par la règle $< X > \rightarrow u < S >$
    4. Garder toutes les autres règles de dérivation de G_α.

Donc

$\longrightarrow$ on partira du nouvel axiome

$\longrightarrow$ soit on génèrera soit le mot vide

$\longrightarrow$ soit on ira à l'axiome de G_α on génèrera un mot associé à α on retournera en 1.

$\longrightarrow$ on pourra générer un mot qui est la concaténation d'un nombre quelconque de mots associés à α.

$\longrightarrow$ on générera le même langage que $(\alpha)^*$.

Grammaire et expression régulière

Conclusion :

- on peut associer un grammaire régulière à chaque expression régulière.
- tout langage est régulier s'il est le langage associé à une expression régulière.
- Un langage régulier est toujours associé à une expression régulière

  $\longrightarrow$ à partir de celle-ci on peut toujours obtenir une grammaire régulière générant le même langage

  $\longrightarrow$ $\boxed{\text{tout langage régulier peut-être engendré par une grammaire régulière.}}$

Grammaire et expression régulière : Exemple

Construisons la grammaire régulière engendrant le langage associé à l'expression régulière

$$\left(ab^* \middle| b^* c\right)^*$$

Ecrivons cette expression régulière comme $(\alpha|\beta)^*$ avec $\alpha = ab^*$ et $\beta = b^* c$. Alors

1) La grammaire G_α associé à α est

$$\begin{array}{lcll} < S_\alpha > & \to & a < B > & \text{la première lettre sera toujours } a \\ < B > & \to & \varepsilon \Big| b < B > & \text{Quand on arrive à } < B > \text{ on a déjà la lettre } a \\ & & & \text{arrêt ou ajout d'un nombre quelconque de } b \end{array}$$

2) La grammaire G_β associé à β est

$$\begin{array}{lcll} < S_\beta > & \to & b < S_\beta > \Big| c & \text{soit prendre un } b \text{ et continuer} \\ & & & \text{soit prendre un } c \text{ et s'arrêter} \end{array}$$

Grammaire et expression régulière : Exemple

3) Donc la grammaire associée à $\gamma = (\alpha|\beta)$ est

$$\begin{array}{rcll}
< S_\gamma > & \rightarrow & < S_\alpha > \Big| < S_\beta > & \text{nouvel axiome } < S_\gamma > \text{ et sa règle} \\
< S_\alpha > & \rightarrow & a < B > & \text{Règle de dérivation de } G_\alpha \\
< B > & \rightarrow & \varepsilon \Big| b < B > & \text{Règle de dérivation de } G_\alpha \\
< S_\beta > & \rightarrow & c \Big| b < S_\beta > & \text{Règle de dérivation de } G_\beta
\end{array}$$

4) Donc la grammaire associée à l'expression régulière $\left(ab^* \Big| b^* c\right)^* = \gamma^*$ est

$$\begin{array}{rcll}
< S > & \rightarrow & \varepsilon \Big| < S_\gamma > & \text{nouvel axiome } < S > \text{ et sa règle} \\
< S_\gamma > & \rightarrow & < S_\alpha > \Big| < S_\beta > & \text{Règle de dérivation de } G_\gamma \\
< S_\alpha > & \rightarrow & a < B > & \text{Règle de dérivation de } G_\alpha \\
< B > & \rightarrow & \color{red}{< \mathbf{S} >} \Big| b < B > & \text{Règle de dérivation de } G_\alpha \text{ } \color{red}{\text{modifiée}} \\
< S_\beta > & \rightarrow & c \color{red}{< \mathbf{S} >} \Big| b < S_\beta > & \text{Règle de dérivation de } G_\beta \text{ } \color{red}{\text{modifiée}}
\end{array}$$

Grammaire et expression régulière : Exemple

Remarque :

G_γ : après être parti de l'axiome $< S_\gamma >$, on ne sait pas y revenir.

$\longrightarrow$ on peut alors garder $< S_\gamma >$ comme axiome

$\longrightarrow$ ajout la règle de dérivation $< S_\gamma > \rightarrow \varepsilon$.

Donc une autre grammaire possible pour $\left(ab^* \middle| b^*c\right)^* = \gamma^*$ est

$$
\begin{array}{rcll}
< S_\gamma > & \rightarrow & \varepsilon \Big| < S_\alpha > \Big| < S_\beta > & \text{Règle de dérivation de } G_\gamma \text{ +autre règle} \\
< S_\alpha > & \rightarrow & a < B > & \text{Règle de dérivation de } G_\alpha \\
< B > & \rightarrow & < \boldsymbol{S_\gamma} > \Big| b < B > & \text{Règle de dérivation de } G_\alpha \text{ modifiée} \\
< S_\beta > & \rightarrow & c < \boldsymbol{S_\gamma} > \Big| b < S_\beta > & \text{Règle de dérivation de } G_\beta \text{ modifiée}
\end{array}
$$

Grammaire et expression régulière : Conclusion

On a montré que **Tout langage décrit par une expression régulière peut être engendré par une grammaire régulière**

On peut aussi montrer (mais on ne le fera pas) que

Tout langage engendré par une grammaire régulière peut être décrit par une expression régulière.

Donc un langage sur un alphabet Σ est régulier

ssi il est le langage associé à une expression régulière sur Σ
ssi il est le langage engendré par une grammaire régulière

Grammaire et expression régulière : Conclusion

Exemple :

Soit le langage L formé de tous les mots sur l'alphabet $\Sigma = \{a,\ b,\ c\}$ contenant un nombre pair de '*a*'.

Ce langage est-il régulier ?

$\longrightarrow$ grammaire régulière pouvant engendrer ce langage ?

Langage "donné en compréhension" $\longrightarrow$ réfléchir en terme d'état.

Etat possible du mot ?

Variation de l'état du mot en fonction des lettres qu'on lui ajoute ?

- On cherche les mots qui ont un nombre pair de '*a*'.

  $\longrightarrow$ l'état va être fonction du nombre de '*a*' :

  $\longrightarrow$ Le mot peut avoir soit un nombre pair de '*a*'
  soit un nombre impair de '*a*'
- Au départ : mot vide

  $\longrightarrow$ nombre pair de '*a*' (0)

  $\longrightarrow$ axiome $< S >$: le mot a un nombre pair de '*a*'
- Ajoutons le symbole non terminal $< A >$: le mot a un nombre impair de '*a*'

Grammaire et expression régulière : Conclusion

Regardons les variations d'états en fonction de l'ajout d'une nouvelle lettre :

1. Si le mot a un nombre pair de *a* alors
   - un mot acceptable car mot du langage.
     $\longrightarrow$ règle de dérivation : $< S > \rightarrow\ \varepsilon$
   - si on ajoute un *a* alors le nombre de *a* devient impair.
     $\longrightarrow$ le mot change d'état
     $\longrightarrow$ la règle de dérivation $< S > \rightarrow\ a < A >$
   - si on ajoute un *b* alors le nombre de *a* ne change pas et reste pair.
     $\longrightarrow$ le mot reste dans le même état
     $\longrightarrow$ règle de dérivation $< S > \rightarrow\ b < S >$
   - si on ajoute un *c* alors le nombre de *a* ne change pas et reste pair.
     $\longrightarrow$ le mot reste dans le même état
     $\longrightarrow$ règle de dérivation $< S > \rightarrow\ c < S >$

   Ces quatres règles peuvent se réécrire

$$< S > \rightarrow\ \varepsilon \,\Big|\, a < A > \,\Big|\, b < S > \,\Big|\, c < S >$$

Grammaire et expression régulière : Conclusion

2. Si le mot a un nombre impair de *a* alors
   - si on ajoute un *a* alors le nombre de *a* devient pair.
     - $\longrightarrow$ le mot change d'état
     - $\longrightarrow$ la règle de dérivation $< A > \rightarrow\ a < S >$
   - si on ajoute un *b* alors le nombre de *a* ne change pas et reste impair.
     - $\longrightarrow$ le mot reste dans le même état
     - $\longrightarrow$ règle de dérivation $< A > \rightarrow\ b < A >$
   - si on ajoute un *c* alors le nombre de *a* ne change pas et reste impair.
     - $\longrightarrow$ le mot reste dans le même état
     - $\longrightarrow$ règle de dérivation $< A > \rightarrow\ c < A >$

Ces trois règles peuvent se réécrire

$$< A > \rightarrow\ a < S > \Big| b < A > \Big| c < A >$$

Grammaire et expression régulière : Conclusion

On a donc trouvé la grammaire suivante qui engendre L :

$$\boxed{\begin{array}{lcl} <S> & \to & \varepsilon \mid a<A> \mid b<S> \mid c<S> \\ <A> & \to & a<S> \mid b<A> \mid c<A> \end{array}}$$

Cette grammaire étant régulière, le langage L est donc régulier.

Expression régulière dont L serait le langage associé ?

Ajout de 'a' toujours par 2 pour garder un nombre pair.

Pour avoir toujour un mot du langage L après ajout de lettre(s) il faut

- soit ajouter un 'b' $\to$ expression régulière : b
- soit ajouter un 'c' $\to$ expression régulière : c
- soit ajouter deux 'a' avec un nombre quelques de 'b' et/ou de 'c' entre les 2 $\to$ expression régulière : $a(b|c)^*a$

Donc une expression régulière dont L est le langage associé est la suivante

$$\boxed{\left(b \middle| c \middle| a(b|c)^*a\right)^*}$$

Automates de Moore et NDFA : Introduction

- But de ce chapitre : déterminer si un langage est régulier (compilable).

- Deux outils permettent de faire cela vu précédemment :
  - les expressions régulières
  - les grammaires régulières.

- Dans cette section : troisième outil : les automates.

Automates de Moore : Définition

Une automate de Moore est composé par :

1) un alphabet Σ (appelé ensemble des **symboles terminaux**)
2) un ensemble fini non vide E dont les éléments sont appelés **états**
3) un élément e_0 de E, appelé **état initial** ;
4) un sous-ensemble A de E, dont les éléments sont appelés **états acceptants** ;
5) une fonction $t : \Sigma \times E \to E$ dite fonction de transition d'états.

Un automate de Moore est donc un quintuplet $\boldsymbol{M} = (\Sigma,\ E,\ e_0, A,\ t)$.

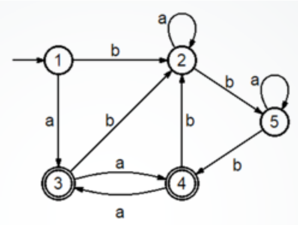
Automates de Moore : Langage Engendré

A partir d'un automate de Moore, on peut engendrer un langage :

Le langage L « reconnu » par $\boldsymbol{M}$ est constitué de tous les mots sur Σ qui, au départ de e_0, conduisent, par transition, à un état final acceptant.

Automates de Moore : Exemple

Soit l'alphabet $\Sigma = \{a,\ b\}$ et l'automate suivant sur Σ



Alors,

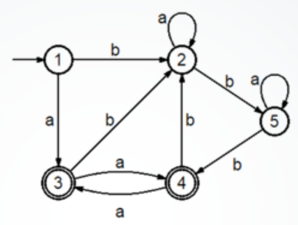
1. L'ensemble des états est $E = \{1,\ 2,\ 3,\ 4,\ 5\}$
2. L'état initial est $e_0 = 1$ (marqué par une $\rightarrow$ ne venant d'aucun état)
3. L'ensemble des états acceptants est $A = \{3,\ 4\}$ (les doubles cercles)
4. La fonction de transition est indiquée par les flèches :
   Une flèche $(i) \xrightarrow{x} (j)$ signifie que, si on est à l'état i et que la lettre suivante du mot est x, alors on va passer à l'état j.

Automates de Moore : Exemple

Un automate est un automate de Moore ssi

- il possède un et seul état entrant
- pour tout état *i* et pour toute lettre de l'alphabet *x*, il existe une et une seule flèche partant de *i* étiquetée avec *x*

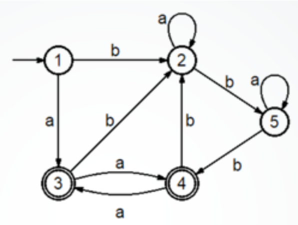
Notre exemple :



- a un seul état entrant : 1
- de chaque état partent deux flèches : une étiquetée *a* et une étiquetée *b*

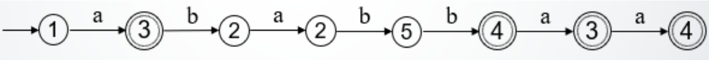
Comme l'alphabet $\Sigma = \{a, b\} \rightarrow$ automate de Moore !

Automates de Moore : Exemple



Exemples de transitions :

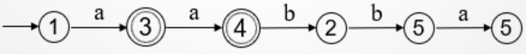
1. Soit la suite de transitions :



   → **Commence à l'état initial et se termine sur l'état 4 qui est acceptant**

   → **le mot** *ababbaa* **appartient au langage reconnu par cet automate**

2. Soit la suite de transitions :



   → **commence à l'état initial et se termine sur l'état** 5 **qui n'est pas acceptant**

   → **le mot** *aabba* **n'appartient au langage reconnu par cet automate**

Automates de Moore : Exemple

Conclusion :

Pour obtenir un mot du langage reconnu par l'automate il faut

- Commencer à l'état entrant
- Passer d'un état à un autre en suivant les flèches
- Terminer sur un état acceptant
- Concaténer, dans l'ordre du parcours des flèches, les lettres indexant ces flèches.

Automates de Moore : Table de transitions

Un automate de Moore peut être entièrement décrit par une table de transition d'états.

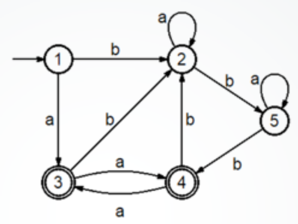
C'est une table à double entrée obtenue en mettant

- les états en en-tête de colonne
- les lettres de l'alphabet Σ en en tête de ligne
- l'état auquel on arrive en partant de l'état i en suivant la flèche indexée de la lettre a dans la case à l'intersection de la ligne correspondant à l'état i et de la colonne correspondant à lettre a

Automates de Moore : Table de transitions

Dans notre exemple, on obtient la table de transition :

Automate de Moore	Table de transitions



	↓					
	1	**2**	**3**	**4**	**5**	***E***
a	3	2	4	3	5	
b	2	5	2	2	4	
Σ						

La table de droite décrit entièrement l'automate de Moore de gauche.
En effet,

- Les états sont en-tête de colonne
- Les lettres de l'alphabet sont en en-tête de ligne
- Les états acceptant sont en rouge
- La flèche ↓ au-dessus du 1 indique que c'est l'état initial (entrant)

Automates de Moore : Table de transitions

Dans notre exemple :

Automate

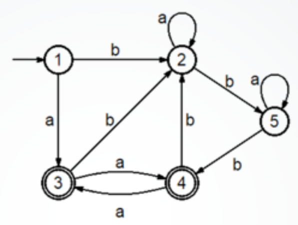


Table de transitions

	1	**2**	**3**	**4**	**5**	***E***
a	3	2	4	3	5	
b	2	5	2	2	4	
Σ						

La table nous dit par exemple que de l'état 1

- si on prend la flèche indexée par *a* on arrivera à l'état 3
- si on prend la flèche indexée par *b* on arrivera à l'état 2

NDFA (Non Deterministic Finite Automaton) : Définition

Un NDFA est composé par :

1) un alphabet Σ (appelé ensemble des **symboles terminaux**)
2) un ensemble fini non vide E dont les éléments sont appelés **états**
3) un sous-ensemble I de E, dont les éléments sont les **états initiaux** ;
4) un sous-ensemble A de E, dont les éléments sont appelés **états acceptants** ;
5) une fonction $t : \Sigma \times E \to E$ dite fonction de transition d'états.

Un NDFA est donc un quintuplet $\boldsymbol{M} = (\Sigma,\ E,\ I, A,\ t)$.

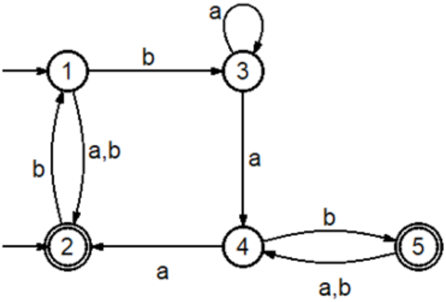
NDFA : Langage Engendré

A partir d'un NDFA, on peut engendrer un langage :

Un mot u est ≪ **reconnu** ≫ par $\boldsymbol{M}$ s'**il existe** une séquence de transitions étiquetées par les lettres de u conduisant d'**un** des états initiaux à un état final d'acceptation.

NDFA : Exemple

Soit l'alphabet $\Sigma = \{a, b\}$ et le NDFA suivant sur Σ

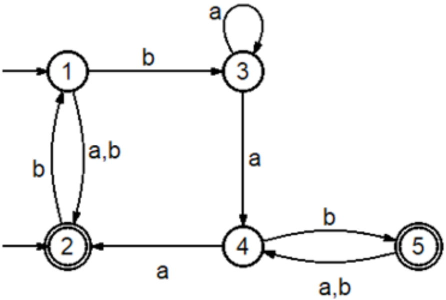


Alors,

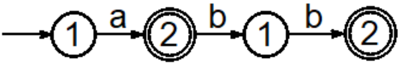
1. L'ensemble des états est $E = \{1, 2, 3, 4, 5\}$
2. L'ensemble des états initiaux est $A = \{1, 2\}$ (marqués par des $\rightarrow$ ne venant d'aucun état)
3. L'ensemble des états acceptants est $A = \{2, 5\}$ (les doubles cercles)

NDFA : Exemple de transitions

Soit l'alphabet $\Sigma = \{a, b\}$ et le NDFA suivant sur Σ

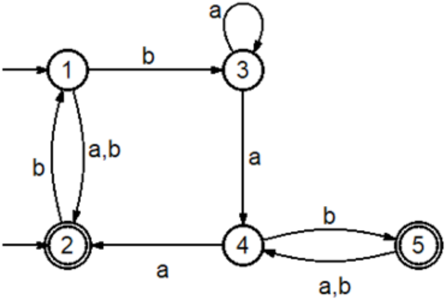


1) Le mot *abb* **est** reconnu car on a la **suite de transitions**



$\rightarrow$ **commence par un des états initiaux et se termine sur un état acceptant**

NDFA : Exemple de transitions

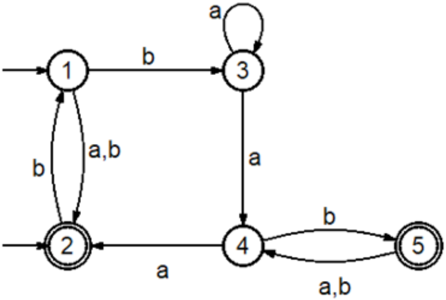


2) Le mot *bbab* **est reconnu** car on a la **suite de transitions** :

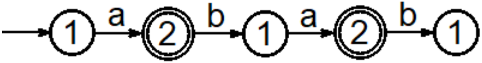
$$\rightarrow 2 \xrightarrow{b} 1 \xrightarrow{b} 3 \xrightarrow{a} 4 \xrightarrow{b} 5$$

$\rightarrow$ **commence par un des états initiaux et se termine sur un état acceptant**

NDFA : Exemple de transitions

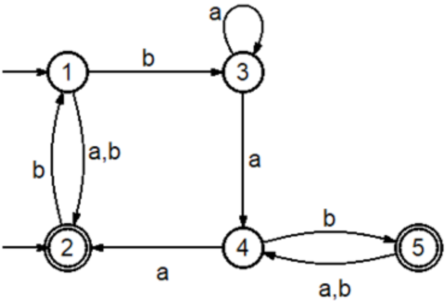


3) Le mot *abab* **n'est pas reconnu**



→ **la seule suite de transitions donnant se mot se termine sur un état qui n'est pas acceptant.**

NDFA : Exemple de transitions



4) Le mot *aab* **n'est pas reconnu**

→ **Il n'y a aucune suite de transition correspondant à ce mot car il est impossible de commencer par deux a.**

NDFA : Conclusion

Un NDFA est automate dans lequel on peut trouver :

- plusieurs états initiaux !
- plusieurs transitions possibles dans une même configuration !
- des configurations dans lesquelles aucune transition n'est prévue !

NDFA : Table de transitions

Un NDFA peut être entièrement décrit par une table de transition d'états.

C'est une table à double entrée obtenue en mettant

- les états en en-tête de colonne
- les lettres de l'alphabet Σ en en tête de ligne
- tous les états auxquels on arrive en partant de l'état i en suivant une flèche indexée de la lettre a dans la case à l'intersection de la ligne correspondant à l'état i et de la colonne correspondant à lettre a

NDFA : Table de transitions

Dans notre exemple, on obtient la table de transition :

NDFA

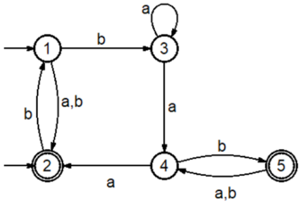


Table de transitions

	$\downarrow$	$\downarrow$				
	1	**2**	**3**	**4**	**5**	E
a	2	—	3, 4	2	4	
b	2, 3	1	—	5	4	
Σ						

La table de droite décrit entièrement le NDFA de gauche.

En effet,

- Les états sont en-tête de colonne
- Les lettres de l'alphabet sont en en-tête de ligne
- Les états acceptant sont en rouge
- Les flèches $\downarrow$ indiquent les états initiaux
- Les — signifient qu'il n'y pas de flèche indexée par la lettre correspondant à la colonne sortant de l'état correspondant à la ligne

NDFA : Remarques

- Un automate de Moore peut être vu comme un cas particulier,de NDFA où
  - l'ensemble I des états initaux est un singleton ;
  - $\forall x \in \Sigma, \forall e \in E : T(x, e)$ est un singleton.
- Quel peut bien être l'intérêt pratique d'un automate NON déterministe ?
  - souvent plus simple à concevoir
  - à partir d'un NDFA, on peut construire, grâce à un algorithme simple, un automate de Moore reconnaissant le même langage !

Subset construction : Description formelle

Algorithme permettant d'obtenir un automate de Moore engendrant le même langage qu'un NDFA

Algorithme :

À partir du NDFA $M = (\Sigma, E, I, A, T)$, on construit l'automate de Moore M' où

- chaque état est un sous-ensemble de E ;
- l'état initial est I (l'ensemble des états initiaux)
- un état est acceptant s'il contient un état acceptant du NDFA
- la fonction de transition τ est définie par : $\tau(x, D) = \bigcup_{e \in D} T(x, e)$

  (ensemble de tous les états auxquels on peut aboutir en suivant une flèche indexée par x à partir d'un état appartenant à D dans le NDFA)

L'automate de Moore ainsi construit est tel que

1) Les langages reconnus par les automates M et M' sont égaux.
2) Le nombre maximum d'états de M' est $2^{|E|}$.

Attention ! L'algorithme *subset construction* ne fournit généralement pas un automate de Moore optimisé !

Subset construction : Exemple

Soit l'alphabet $\{a,\ b\}$ et le NDFA suivant sur cet alphabet avec sa table de transition

NDFA

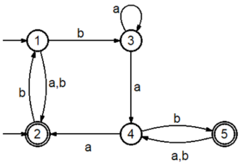
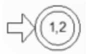


Table de transitions

	$\downarrow$	$\downarrow$				
	1	**2**	**3**	**4**	**5**	E
a	2	—	3, 4	2	4	
b	2, 3	1	—	5	4	
Σ						

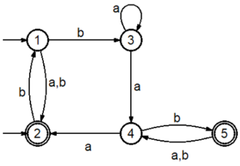
Construisons un automate de Moore générant le même langage :

1) L'état initial est $\{1,\ 2\}$, l'ensemble des états initiaux du NDFA.
   $\rightarrow$ état acceptant car 2 est acceptant :



Subset construction : Exemple

NDFA	Table de transitions

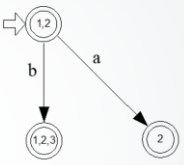


	↓ 1	↓ 2	3	4	5	*E*
a	2	—	3, 4	2	4	
b	2, 3	1	—	5	4	
Σ						

2) Etat $\{1, 2\}$: avec le NDFA, en partant des états 1 et 2 et :

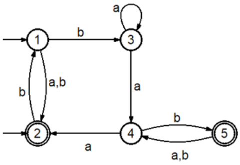
- en choisissant *a* comme caractère suivant, on ne peut aller qu'à l'état 2 $\rightarrow$ flèche indexée par *a* vers le nouvel l'état $\{2\}$ acceptant car 2 l'est ;
- en choisissant *b* comme caractère suivant, on peut aller aux états $1,\ 2$ et 3 $\rightarrow$ flèche indexée par *b* vers le nouvel état $\{1,\ 2,\ 3\}$ acceptant car 2 l'est.

On obtient :



Subset construction : Exemple

NDFA	Table de transitions

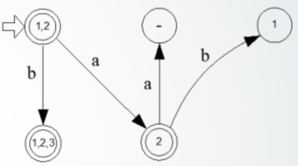


	$\downarrow$	$\downarrow$				
	1	**2**	**3**	**4**	**5**	***E***
a	2	—	3, 4	2	4	
b	2, 3	1	—	5	4	
Σ						

3) Etat $\{2\}$: avec le NDFA, en partant de l'état 2 et :

- en choisissant *a* comme caractère suivant, on ne peut aller nulle part $\rightarrow$ flèche indexée par *a* vers le nouvel état $-$, état poubelle **non acceptant**
- en choisissant *b* comme caractère suivant, on peut aller qu'à l'état 1 $\rightarrow$ flèche indexée par *b* vers le nouvel état $\{1\}$ non acceptant car 1 ne l'est pas.

On obtient :



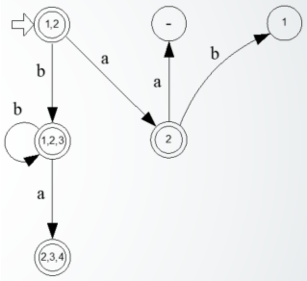
Subset construction : Exemple

4) Etat $\{1, 2, 3\}$: avec le NDFA, en partant des états 1, 2 et 3 et :
   - en choisissant *a* comme caractère suivant, on peut aller aux états 2, 3 et 4 $\rightarrow$ flèche indexée par *a* vers le nouvel état $\{2, 3, 4\}$ acceptant car 2 l'est ;
   - en choisissant *b* comme caractère suivant, on peut aller aux états 1, 2 et 3 $\rightarrow$ flèche indexée par *b* de l'état $\{1, 2, 3\}$ vers lui-même

Table de transitions

Σ \ E	$\downarrow$ **1**	$\downarrow$ **2**	**3**	**4**	**5**
a	2	—	3, 4	2	4
b	2, 3	1	—	5	4

Automate de Moore obtenu



Subset construction : Exemple

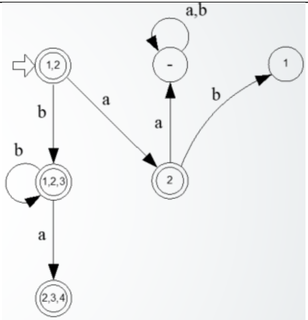
5) Regardons l'état poubelle $-$:

- C'est un état "sans issue"
  $\rightarrow$ flèche indexée par *a* et *b* de cet état vers lui-même.

Table de transitions

	↓ 1	↓ 2	3	4	5	*E*
a	2	$-$	3, 4	2	4	
b	2, 3	1	$-$	5	4	
Σ						

Automate de Moore obtenu



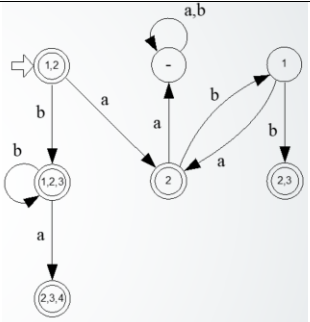
Subset construction : Exemple

6) Etat $\{1\}$: avec le NDFA, en partant de l'état 1 et :
   - en choisissant *a* comme caractère suivant, on peut aller qu'à l'état 2
     $\rightarrow$ flèche indexée par *a* vers l'état $\{2\}$
   - en choisissant *b* comme caractère suivant, on peut aller aux états 2 et 3
     $\rightarrow$ flèche indexée par *b* vers le nouvel état $\{2, 3\}$ acceptant car 2 l'est.

Table de transitions

	$\downarrow$ **1**	$\downarrow$ **2**	**3**	**4**	**5**	***E***
a	2	$-$	$3,4$	2	4	
b	$2,3$	1	$-$	5	4	
Σ						

Automate de Moore obtenu



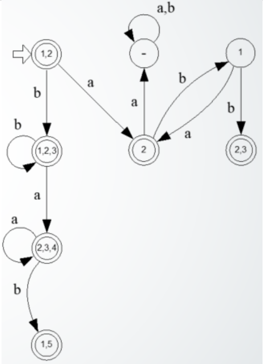
Subset construction : Exemple

7) Etat $\{2, 3, 4\}$: avec le NDFA, en partant des états 2, 3 et 4 et :
   - en choisissant *a* comme caractère suivant, on peut aller aux états 2, 3 et 4
     $\rightarrow$ flèche indexée par *a* vers de l'état $\{2, 3, 4\}$ vers lui-même
   - en choisissant *b* comme caractère suivant, on peut aller aux états 1 et 5
     $\rightarrow$ flèche indexée par *b* vers le nouvel état $\{1, 5\}$ acceptant car 5 l'est.

Table de transitions

	↓ **1**	↓ **2**	**3**	**4**	**5**	***E***
a	2	—	3, 4	2	4	
b	2, 3	1	—	5	4	
Σ						

Automate de Moore obtenu



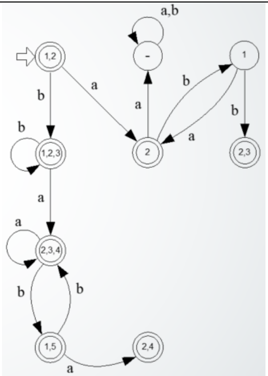
Subset construction : Exemple

8) Etat $\{1,5\}$: avec le NDFA, en partant des états 1 et 5 et :
   - en choisissant *a* comme caractère suivant, on peut aller aux états 2 et 4
     $\rightarrow$ flèche indexée par *a* vers le nouvel état $\{2,4\}$ acceptant car 2 l'est.
   - en choisissant *b* comme caractère suivant, on peut aller aux états 2, 3 et 4
     $\rightarrow$ flèche indexée par *b* vers l'état $\{2,\ 3,\ 4\}$

Table de transitions

Σ \ E	$\downarrow$ 1	$\downarrow$ 2	3	4	5
a	2	—	3, 4	2	4
b	2, 3	1	—	5	4

Automate de Moore obtenu



Subset construction : Exemple

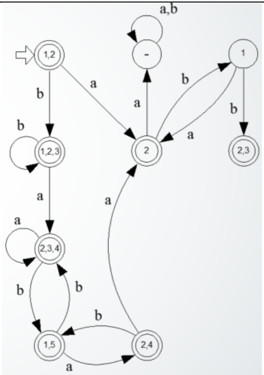
9) Etat $\{2,4\}$: avec le NDFA, en partant des états 2 et 4 et :

- en choisissant *a* comme caractère suivant, on peut aller qu'à l'état 2
  $\rightarrow$ flèche indexée par *a* vers l'état $\{2\}$
- en choisissant *b* comme caractère suivant, on peut aller aux états 1 et 5
  $\rightarrow$ flèche indexée par *b* vers l'état $\{1,\ 5\}$

Table de transitions

Σ \ E	1	2	3	4	5
a	2	—	3, 4	2	4
b	2, 3	1	—	5	4

Automate de Moore obtenu

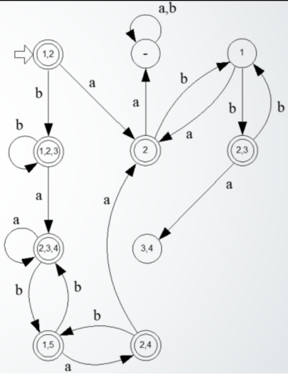


Subset construction : Exemple

10) Etat $\{2, 3\}$ non traité : avec le NDFA, en partant des états 2 et 3 et :
   - en choisissant *a* comme caractère suivant, on peut aller aux états 3 et 4
     $\rightarrow$ flèche indexée par *a* vers l'état $\{3,\ 4\}$ non acceptant (ni 3 ni 4 ne le sont)
   - en choisissant *b* comme caractère suivant, on peut aller qu'à l'état 1
     $\rightarrow$ flèche indexée par *b* vers l'état $\{1\}$

Table de transitions	Automate de Moore obtenu

	$\downarrow$ 1	$\downarrow$ 2	3	4	5	***E***
a	2	—	3, 4	2	4	
b	2, 3	1	—	5	4	
$\boldsymbol{\Sigma}$						

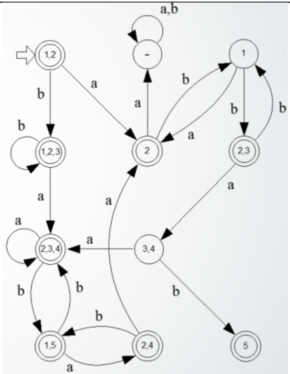


Subset construction : Exemple

11) Etat $\{3, 4\}$: avec le NDFA, en partant des états 3 et 4 et :
   - en choisissant *a* comme caractère suivant, on peut aller aux états 2, 3 et 4
     $\rightarrow$ flèche indexée par *a* vers l'état $\{2,\ 3,\ 4\}$
   - en choisissant *b* comme caractère suivant, on peut aller qu'à l'état 5
     $\rightarrow$ flèche indexée par *b* vers le nouvel état $\{5\}$ acceptant car 5 l'est

Table de transitions	Automate de Moore obtenu

	↓	↓				
	1	**2**	**3**	**4**	**5**	***E***
a	2	—	3, 4	2	4	
b	2, 3	1	—	5	4	
Σ						



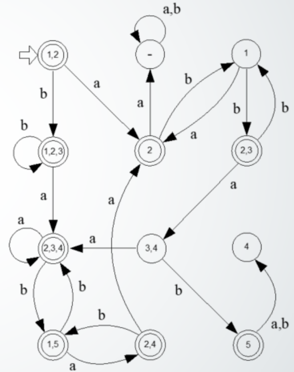
Subset construction : Exemple

12) Etat $\{5\}$: avec le NDFA, en partant de l'état 5 et :
   - en choisissant *a* comme caractère suivant, on peut aller qu'à l'état 4
     $\rightarrow$ flèche indexée par *a* vers le nouvel état $\{4\}$ non acceptant comme 4
   - en choisissant *b* comme caractère suivant, on peut aller qu'à l'état 4
     $\rightarrow$ flèche indexée par *b* vers l'état $\{4\}$

Table de transitions

	↓ 1	↓ 2	3	4	5	*E*
a	2	—	3, 4	2	4	
b	2, 3	1	—	5	4	
Σ						

Automate de Moore obtenu



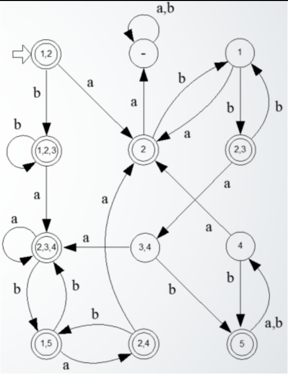
Subset construction : Exemple

13) Etat $\{4\}$: avec le NDFA, en partant de l'état 4 et :

- en choisissant *a* comme caractère suivant, on peut aller qu'à l'état 2
  $\rightarrow$ flèche indexée par *a* vers l'état $\{2\}$
- en choisissant *b* comme caractère suivant, on peut aller qu'à l'état 5
  $\rightarrow$ flèche indexée par *b* vers l'état $\{5\}$

Table de transitions	**Automate de Moore obtenu**

	$\downarrow$ **1**	$\downarrow$ **2**	**3**	**4**	**5**	***E***
a	2	—	3, 4	2	4	
b	2, 3	1	—	5	4	
Σ						

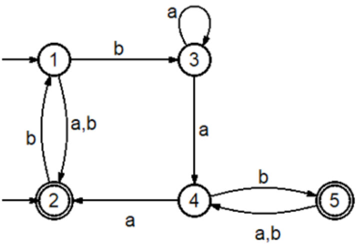
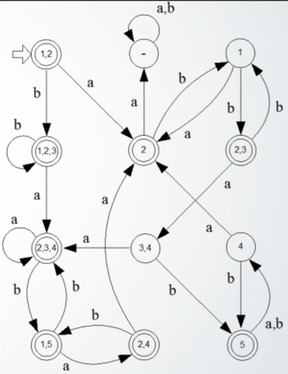


Subset construction : Exemple

Il n'y a plus d'état à traiter !

⟶ L'algorithme est terminé !

⟶ L'automate obtenu en 13, est un automate de Moore reconnaissant le même langage que le NDFA de départ !

NDFA	Automate de Moore obtenu
	

Subset construction : Exemple

L'algorithme de la subset construction est donc

1) Prendre l'ensemble des états initiaux du NDFA comme état initial de l'automate de Moore correspondant

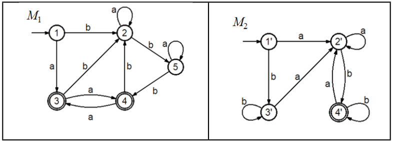
2) S'il reste un état *e* non traité :

   a) Pour chaque lettre de l'alphabet, faire une flèche indexée par cette lettre de cet état vers l'état *f* qui sera l'ensemble de tous les états du NDFA auxquels on peut arriver en suivant une flèche indexée par cette lettre partant d'un état du NDFA appartenant à l'état *e*.

   b) Recommencer en 2.

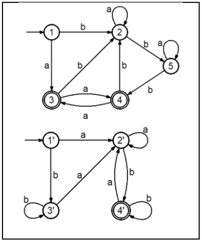
Automate de Moore pour l'union de 2 langages

Soient L_1, L_2 2 langages reconnus par les automates de Moore M_1 et M_2 :



Pour trouver un automate de Moore engendrant le langage $L_1 \cup L_2$, il faut

1) Rassembler les deux automates M_1 et M_2 en un seul NDFA.



2) Appliquer la *subset construction* au NDFA obtenu en 1.

Automate de Moore pour la concaténation de 2 langages

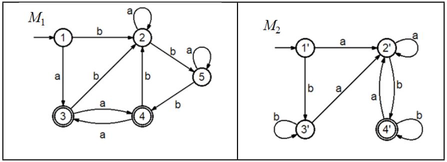
Soient L_1, L_2 2 langages reconnus par les automates de Moore M_1 et M_2 :

Étapes pour obtenir un automate de Moore engendrant le langage $L_1 \bullet L_2$:

1. Mettre les deux automates M_1 et M_2 ensemble
2. Pour toute flèche étiquetée x d'un état i vers un état j où j est un état acceptant de M_1, on ajoute une flèche étiquetée par x de l'état i vers l'état initial de M_2.
3. Si l'état initial de M_1 n'est pas acceptant, alors retirer l'état initial de M_2 des états initiaux du NDFA obtenu en 2.
4. Si l'état initial de M_2 n'est pas acceptant, alors rendre non acceptant tous les états acceptants de M_1
5. Appliquer la *subset construction* au NDFA obtenu en 4.

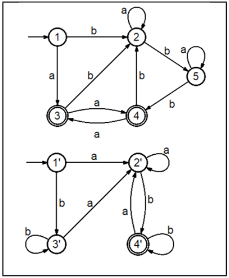
Conacténation de 2 langages : Exemple

Soient L_1, L_2 2 langages reconnus par les automates de Moore M_1 et M_2 :



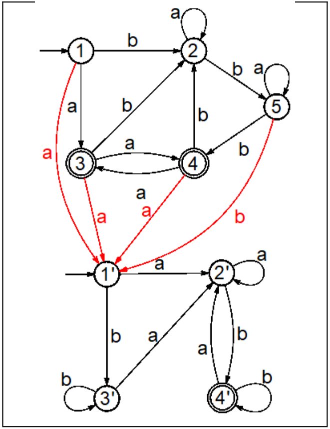
Trouvons un automate de Moore reconnaissant le langage $L_1 \bullet L_2$

1) Rassemblons les deux automates M_1 et M_2 en un seul NDFA.



Conacténation de 2 langages : Exemple

3) Pour toute flèche étiquetée x d'un état i vers un état j où j est un état acceptant de M_1, ajoutons une flèche étiquetée par x de l'état i vers l'état initial de M_2.



ajout de $1 \xrightarrow{a} 1'$ car $1 \xrightarrow{a} 3$ acceptant dans M_1

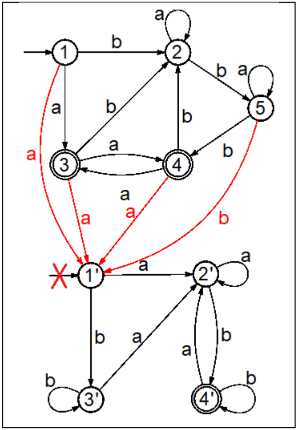
ajout de $3 \xrightarrow{a} 1'$ car $3 \xrightarrow{a} 4$ acceptant dans M_1

ajout de $4 \xrightarrow{a} 1'$ car $4 \xrightarrow{a} 3$ acceptant dans M_1

ajout de $5 \xrightarrow{b} 1'$ car $5 \xrightarrow{b} 4$ acceptant dans M_1

Conacténation de 2 langages : Exemple

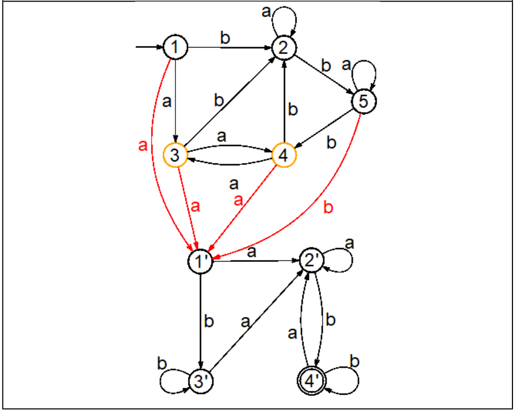
4) Retirons l'état $1'$ comme état initial car l'état initial 1 de M_1 n'est pas acceptant.



Conacténation de 2 langages : Exemple

4) Rendons non acceptant tous les états acceptant de M_1 car l'état initial de M_2 n'est pas acceptant.

NDFA reconnaissant le langage $L_1 \bullet L_2$



5) Appliquons la *subset construction* au NDFA obtenu en 4.

Automate de Moore pour l'étoile d'un langage

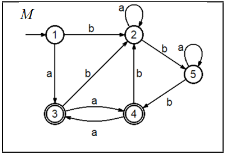
Soit L, un langage reconnu par l'automate de Moore M :

Étape pour obtenir un automate de Moore engendrant le langage L^* :

1. Ajout d'un état acceptant e_i qui sera le nouvel état initial
2. Pour toute flèche étiquetée x de l'**état initial de *M*** vers un état j, on ajoute une flèche étiquetée par x du nouvel état e_i vers l'état j.
3. Pour toute flèche étiquetée x d'un état i vers j un état acceptant de M, on ajoute une flèche étiquetée par x de l'état i vers l'état le nouvel état initial e_i.
4. Retirer l'état initial de M des états initiaux du NDFA obtenu en 3.
5. Appliquer la *subset construction* au NDFA obtenu en 4.

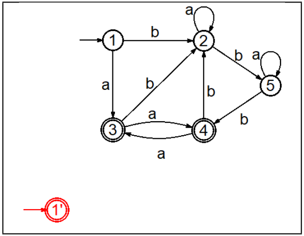
Étoile d'un langage : Exemple

Soient L un langage reconnu par l'automate de Moore M



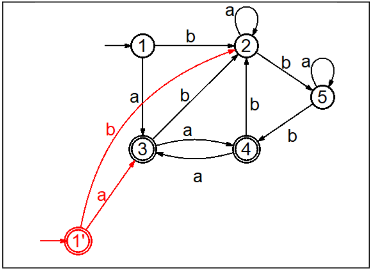
Trouvons un automate de Moore engendrant le langage L^*

1) Ajoutons le nouvel état initial **1'** :



Étoile d'un langage : Exemple

2) Pour toute flèche étiquetée par x de l'**état initial de *M*** vers un état j, on ajoute une flèche étiquetée par x **du nouvel état initial e_i** vers l'état j.

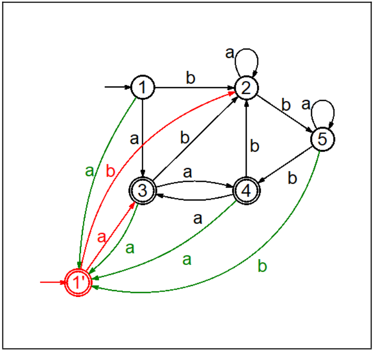


ajout de $1' \xrightarrow{b} 2$ car $1 \xrightarrow{b} 2$ (1 état initial de M)

ajout de $1' \xrightarrow{a} 3$ car $1 \xrightarrow{a} 3$ (1 état initial de M)

Étoile d'un langage : Exemple

3) Pour toute flèche étiquetée x d'un état i vers un état j où j est **un état acceptant de** M, ajoutons une flèche étiquetée par x de l'état, i vers le **nouvel état initial** $1'$.



ajout de $1 \xrightarrow{a} 1'$ car $1 \xrightarrow{a} 3$ (acceptant dans M)

ajout de $3 \xrightarrow{a} 1'$ car $3 \xrightarrow{a} 4$ (acceptant dans M)

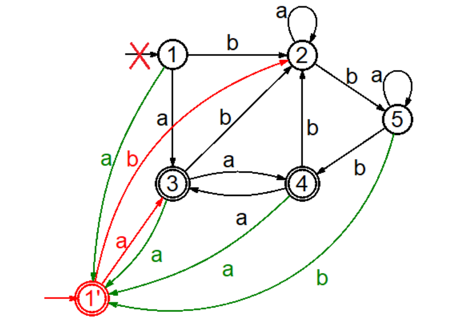
ajout de $4 \xrightarrow{a} 1'$ car $4 \xrightarrow{a} 3$ (acceptant dans M)

ajout de $5 \xrightarrow{b} 1'$ car $5 \xrightarrow{b} 4$ (acceptant dans M)

Étoile d'un langage : Exemple

4) Retirons l'état 1 (l'état initial de M)) comme état initial.

NDFA reconnaissant le langage L^*



5) Appliquer la *subset construction* au NDFA obtenu en 4.

Grammaire associée à un Automate de Moore

Pour tout automate de Moore, on peut trouver une grammaire régulière normalisée générant le langage reconnu par l'automate.

Pour ce faire il faut

1) Associer à chaque état *i* de l'automate un symbole non terminal $< i >$.

2) Associer l'axiome avec l'état initial de l'automate

3) Pour toute flèche indexée par une lettre *x* d'un état *i* vers un état *j*,

   ajouter une règle de dérivation : $< i > \rightarrow\; x < j >$

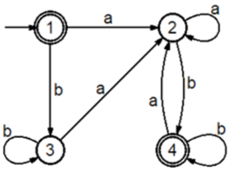
4) Pour tout état acceptant *i*, ajouter la règle de dérivation $< i > \rightarrow\; \varepsilon$

Conclusion :

Les langages reconnus par les automates de Moore sont réguliers

Grammaire associée à un Automate de Moore : Exemple

Soit le langage reconnu par l'automate Moore suivant

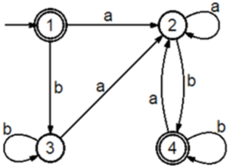


Construisons la grammaire engendrant ce langage.

1) Il y a 4 états : 1, 2, 3 et 4
$\rightarrow$ les symboles non terminaux sont donc $< 1 >$, $< 2 >$, $< 3 >$ et $< 4 >$

2) Le symbole non terminal $< 1 >$ est l'axiome

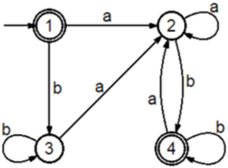
Grammaire associée à un Automate de Moore : Exemple



3) Ajoutons une règle de dérivation par flèche

a) $1 \xrightarrow{a} 2 \Longrightarrow$ règle de dérivation $<1> \rightarrow a <2>$
b) $1 \xrightarrow{b} 3 \Longrightarrow$ règle de dérivation $<1> \rightarrow b <3>$
c) $2 \xrightarrow{a} 2 \Longrightarrow$ règle de dérivation $<2> \rightarrow a <2>$
d) $2 \xrightarrow{b} 4 \Longrightarrow$ règle de dérivation $<2> \rightarrow b <4>$
e) $3 \xrightarrow{a} 2 \Longrightarrow$ règle de dérivation $<3> \rightarrow a <2>$
f) $3 \xrightarrow{b} 3 \Longrightarrow$ règle de dérivation $<3> \rightarrow b <3>$
g) $4 \xrightarrow{a} 2 \Longrightarrow$ règle de dérivation $<4> \rightarrow a <2>$
h) $4 \xrightarrow{b} 4 \Longrightarrow$ règle de dérivation $<4> \rightarrow b <4>$

Grammaire associée à un Automate de Moore : Exemple



4) Ajoutons une règle par état acceptant
   a) l'état 1 est acceptant $\Longrightarrow$ règle de dérivation $<1> \to \varepsilon$
   b) l'état 4 est acceptant $\Longrightarrow$ règle de dérivation $<4> \to \varepsilon$

5) Toutes ces règles mises ensemble donne la grammaire recherchée :

$$\begin{array}{|lcl|l|l|}
\hline
<1> & \to & a<2> & b<3> & \varepsilon \\
<2> & \to & a<2> & b<4> & \\
<3> & \to & a<2> & b<3> & \\
<4> & \to & a<2> & b<4> & \varepsilon \\
\hline
\end{array}$$

Automate de Moore associé à une grammaire régulière normalisée

Pour toute grammaire régulière normalisé, on peut construire un automate de Moore reconnaissance le langage généré par celle-ci.

Pour ce faire il faut

1) Construire un NDFA qui acceptera le langage généré par la grammaire :
   a) Pour chaque symbole non terminal $< I >$ ajouter un état I à l'automate
   b) Prendre l'état associé à l'axiome comme état initial.
   c) Pour toute règle de la forme $< I > \rightarrow \ x < J >$, ajouter $I \xrightarrow{x} J$
   d) Pour toute règle de la forme $< I > \rightarrow \ \varepsilon$, rendre l'état I acceptant.
   e) Pour toute règle de dérivation de la forme $< I > \rightarrow \ x$, avec $x \in \Sigma$, ajouter une flèche indexée par x vers l'état acceptant "universel" Ω (en ajoutant cet état si celui-ci n'est pas déjà présent)
2) Appliquer la subset construction au NDFA construit en 1.

Conclusion :

Les langages réguliers peuvent être reconnus par des automates de Moore

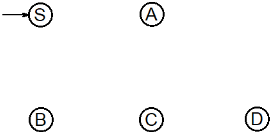
Automate de Moore associé à une grammaire régulière normalisée : exemple

Soit l'alphabet $\Sigma = \{a,\ b,\ c\}$ et la grammaire régulière normalisée suivante

$$\begin{array}{lcl}
< S > & \rightarrow & a < A > \mid b < B > \\
< A > & \rightarrow & b \mid c < S > \\
< B > & \rightarrow & a \mid b < C > \\
< C > & \rightarrow & c < D > \mid \varepsilon \\
< D > & \rightarrow & b < B >
\end{array}$$

Construisons le NDFA générant le langage reconnu par cette grammaire :

1) 5 symboles non terminaux : $< S >$, $< A >$, $< B >$, $< C >$ et $< D >$
   $\Longrightarrow$ l'automate aura comme états S, A, B, C et D.
   $\Longrightarrow$ S est l'état initial car axiome de la grammaire



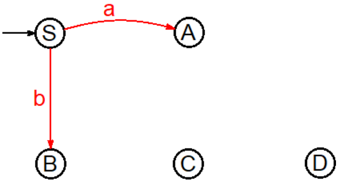
Automate de Moore associé à une grammaire régulière normalisée : exemple

$$\begin{array}{lcl} <S> & \rightarrow & a<A> \mid b<B> \\ <A> & \rightarrow & b \mid c<S> \\ <B> & \rightarrow & a \mid b<C> \\ <C> & \rightarrow & c<D> \mid \varepsilon \\ <D> & \rightarrow & b<B> \end{array}$$

3) Règle de dérivation $<S> \rightarrow a<A> \mid b<B>$

$\Longrightarrow$ ajout de $S \xrightarrow{a} A$.

$\Longrightarrow$ ajout de $S \xrightarrow{b} B$.



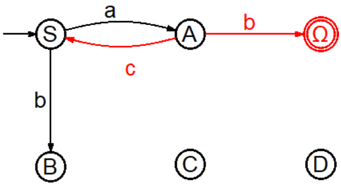
Automate de Moore associé à une grammaire régulière normalisée : exemple

$$
\begin{array}{lcl}
< S > & \to & a < A > \mid b < B > \\
< A > & \to & b \mid c < S > \\
< B > & \to & a \mid b < C > \\
< C > & \to & c < D > \mid \varepsilon \\
< D > & \to & b < B >
\end{array}
$$

4) Règle de dérivation $< A > \to b \mid c < S >$

$\Longrightarrow$ ajout de $A \xrightarrow{b} \Omega$ (l'état universel acceptant qu'on ajoute).

$\Longrightarrow$ ajout de $A \xrightarrow{c} S$.



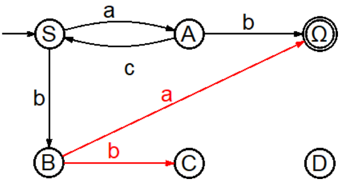
Automate de Moore associé à une grammaire régulière normalisée : exemple

$$
\begin{array}{lcl}
<S> & \rightarrow & a<A> \;|\; b<B> \\
<A> & \rightarrow & b \;|\; c<S> \\
<B> & \rightarrow & a \;|\; b<C> \\
<C> & \rightarrow & c<D> \;|\; \varepsilon \\
<D> & \rightarrow & b<B>
\end{array}
$$

5) Règle de dérivation $<B> \rightarrow a \;|\; b<C>$

$\Longrightarrow$ ajout de $B \xrightarrow{a} \Omega$ (l'état universel) .

$\Longrightarrow$ ajout de $B \xrightarrow{b} C$.



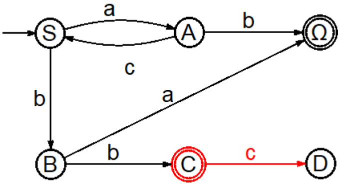
Automate de Moore associé à une grammaire régulière normalisée : exemple

$$
\begin{array}{|lcll|}
\hline
< S > & \rightarrow & a < A > & \mid b < B > \\
< A > & \rightarrow & b \mid c < S > & \\
< B > & \rightarrow & a \mid b < C > & \\
< C > & \rightarrow & c < D > & \mid \varepsilon \\
< D > & \rightarrow & b < B > & \\
\hline
\end{array}
$$

5) Règle de dérivation $< C > \rightarrow c < D > \mid \varepsilon$

$\Longrightarrow$ ajout de $C \xrightarrow{c} D$

$\Longrightarrow$ l'état C devient acceptant.

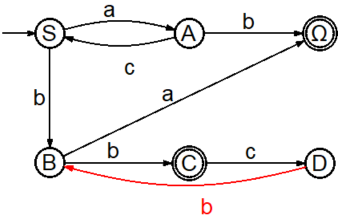


Automate de Moore associé à une grammaire régulière normalisée : exemple

$$
\begin{array}{lcl}
< S > & \rightarrow & a < A > \mid b < B > \\
< A > & \rightarrow & b \mid c < S > \\
< B > & \rightarrow & a \mid b < C > \\
< C > & \rightarrow & c < D > \mid \varepsilon \\
< D > & \rightarrow & b < B >
\end{array}
$$

6) Règle de dérivation $< D > \rightarrow b < B >$

$\Longrightarrow$ ajout de $D \xrightarrow{b} B$



Automate de Moore associé à une grammaire régulière normalisée : conclusion

Pour obtenir un automate de Moore reconnaissant le langage généré par la grammaire
$\rightarrow$ Appliquer la *subset construction* au NDFA obtenu en 6.

Conclusion :

Un langage L sur un alphabet Σ est **régulier**

ssi L est associé à une **expression régulière**

ssi L peut être engendré par une **grammaire régulière**

ssi L peut être engendré par une **grammaire régulière normalisée**

ssi L peut être reconnu par un **automate de Moore**

ssi L peut être reconnu par un **NDFA**

Remarque

Il existe des langages qui ne sont pas réguliers !

<u>Exemple :</u>

Sur l'alphabet $\Sigma = \{a,\ b\}$, le langage des mots ayant le même nombre de *a* et de *b* n'est pas régulier !

En effet, si on essaie de réfléchir en terme d'état.
On pourrait prendre comme état

- Etat 1 : Le nombre de *a* égal le nombre de *b*
- Etat 2 : Le nombre de *a* n'est pas égal au nombre de *b*.

Comment savoir quand on passe de l'état 2 à l'état 1 ?

$\Longrightarrow$ il faudrait toujours connaître la différence entre le nombre de *a* et le nombre de *b* pour savoir quand on revient à l'égalité !

$\Longrightarrow$ il y a une infinité de cas ! !

$\Longrightarrow$ il faudrait une infinité d'états ! ! !

$\Longrightarrow$ le langage n'est pas régulier !

Exercice 1 : Grammaire et Automate de Moore

Soit l'alphabet $\Sigma = \{a,\ b\}$.
Donnez une grammaire régulière et un automate de Moore pour le langage des mots commençant et terminant par *a*, contentant au moins un *b* et ne contenant pas deux *a* consécutifs.

Exercice 1 : Grammaire et Automate de Moore

Les différents états à considérer sont les suivants :

- Mot vide : $< S >$
- Le mot commence par a : $< A >$
- Le mot commence par a et termine par b (donc contient au moins un b) : $< B >$
- Le mot commence par a, contient au moins un b et termine par a : $< C >$

On a alors la grammaire

$$\boxed{\begin{array}{lcl} < S > & \longrightarrow & a < A > \\ < A > & \longrightarrow & b < B > \\ < B > & \longrightarrow & a < C > \big| b < B > \\ < C > & \longrightarrow & \varepsilon \big| b < B > \end{array}}$$

Exercice 1 : Grammaire et Automate de Moore

$$
\begin{array}{lcl}
< S > & \longrightarrow & a < A > \\
< A > & \longrightarrow & b < B > \\
< B > & \longrightarrow & a < C > \mid b < B > \\
< C > & \longrightarrow & \varepsilon \mid b < B >
\end{array}
$$

On a alors l'automate

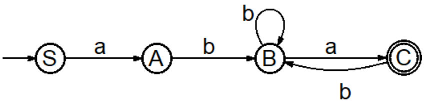


table de transitions

	$\rightarrow S$	A	B	C
a	A	$-$	C	$-$
b	$-$	B	B	B

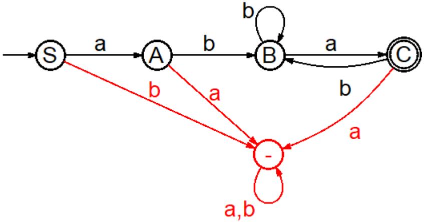
$\Longrightarrow$ C'est un NDFA et pas un automate de Moore

$\Longrightarrow$ Il n'y a qu'un seul état entrant et pas 2 flèches avec la même lettre sortant d'un même état

$\Longrightarrow$ Ajout de l'état poubelle

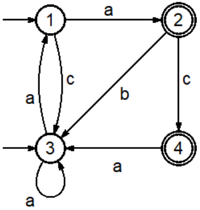
Exercice 1 : Grammaire et Automate de Moore

Automate de Moore



Exercice 2 : Subset construction (Septembre 2019)

Trouvez un automate de Moore reconnaissant le même langage que le NDFA ci-dessous.



Faisons la table de transitions :

	$\rightarrow 1$	2	$\rightarrow 3$	4
a	2	—	$1, 3$	3
b	—	3	—	—
c	3	4	—	—

$\Longrightarrow$ Deux états entrant et deux flèches a sortant de l'état 3

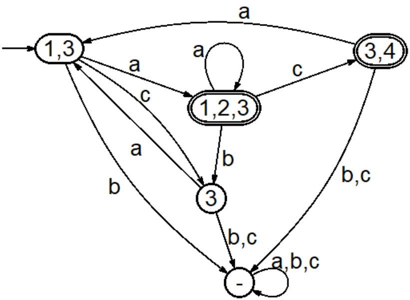
$\Longrightarrow$ Subset construction !

Exercice 2 : Subset construction (Septembre 2019)

Table de transitions :

	$\rightarrow 1$	2	$\rightarrow 3$	4
a	2	—	$1, 3$	3
b	—	3	—	—
c	3	4	—	—

Automate de Moore obtenu par subset construction :



Unicité de l'automate de Moore minimal

Tout langage régulier L est reconnu par UN ET UN SEUL automate de Moore $\boldsymbol{M_L}$ possédant un nombre minimum d'états

Parmi tous les automates de Moore reconnaissant un même langage celui possédant le minimum d'états est appelé automate **optimisé** ou **minimal**.

On a alors le corollaire au théorème précédent :

Si deux automates de Moore distincts M_1 et M_2 ont le même nombre d'états et reconnaissent le même langage alors ces automates ne sont pas optimisés.

Deux automates de Moore sont distincts

- soit s'ils n'ont pas le même nombre d'états
- soit s'ils ont le même nombre d'états mais qu'on ne peut pas passer de l'un à l'autre par un simple renommage d'états

Algorithme de Moore-Nerode : Description formelle

Cet algorithme

1. va chercher les états équivalents afin de les fusionner.

2. va partir d'une équivalence sur l'ensemble des états qui va le partitionner grâce au quotient de celui-ci par cette relation.

3. va raffiner la relation de départ :

   à l'étape *n* :

   va construire une relation équivalence $\sim_n$
   à partir de $\sim_{n-1}$ la relation construite à l'étape $n-1$.

Algorithme de Moore-Nerode : Description formelle

L'algorithme est le suivant.

Etape 0 :

Considérer, sur l'ensemble E des états, la relation d'équivalence $\sim_0$ définie par

$e \sim_0 f$ ssi soit e et f sont tous deux acceptants
soit ni e ni f n'est acceptant.

$\Rightarrow$ partition

$$E/\sim_0 = \Big\{\{\text{ensemble des états acceptants}\}, \{\text{ensemble des états non acceptants}\}\Big\}$$

Etape n :

On définit la relation d'équivalence $\sim_n$ définie par

$e \sim_n f$ ssi $e \sim_{n-1} f$ et pour toute lettre x les flèches indexées par x
sortant des états e et f arrivent sur des états e_2 et f_2
tels que $e_2 \sim_{n-1} f_2$.

Cette relation d'équivalence est appelée équivalence de Nerode.

Algorithme de Moore-Nerode : Description formelle

Les relations de Nerode ont les propriétés suivantes :

- $\neg\left(e \sim_{n-1} f\right) \Rightarrow \neg\left(e \sim_{n} f\right)$
- $\left(e \sim_{n} f\right) \Rightarrow \left(e \sim_{n-1} f\right)$

$\Rightarrow$ Pour obtenir les classes d'équivalence de $\sim_n$ on va partir de celles de $\sim_{n-1}$ et en partitionner certaines.

Critère d'arrêt :

On s'arrête à l'étape n si $\sim_n = \sim_{n-1}$.

Etats à fusionner :

- Si arrêt à l'étape n : fusion de 2 états e et f en un seul ssi $e \sim_{n-1} f$.
- Si les états fusionnés étaient acceptant alors le nouvel état issu de cette fusion est acceptant.
- Si deux états sont fusionnés, alors les flèches qui entraient/sortaient de ces états vont entrer/sortir de l'état issu de cette fusion dans l'automate optimisé.

Algorithme de Moore-Nerode : Exemple

Soit l'automate de Moore suivant et sa table de transitions

Automate de Moore

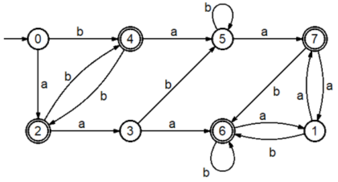


Table de transitions

	↓ 0	1	2	3	4	5	6	7	*E*
a	2	7	3	6	5	7	1	1	
b	4	6	4	5	2	5	6	6	
Σ									

Utilisons l'algorithme de Moore-Nerode pour optimiser cet automate de Moore :

Etape 0 :

On considère $\sim_0$ qui répartit les états en 2 ensembles : les acceptants et les autres.

$\Longrightarrow$ partition $E/\sim_0 = \Big\{[0] = \{0,\ 1,\ 3,\ 5\}\ ,\ [2] = \{2,\ 4,\ 6,\ 7\}\Big\}$

Algorithme de Moore-Nerode : Exemple

	↓ 0	1	2	3	4	5	6	7	E
a	2	7	3	6	5	7	1	1	
b	4	6	4	5	2	5	6	6	
Σ									

Etape 1 : Construction de $\sim_1$ à partir de $\sim_0$:

Regardons chaque classe d'équivalence séparément :

a) Regardons les états de la classe $[0]$:

- Etat 0 : $\begin{cases} 0 \xrightarrow{a} 2 \in [2] \\ 0 \xrightarrow{b} 4 \in [2] \end{cases}$ Etat 1 : $\begin{cases} 1 \xrightarrow{a} 7 \in [2] \\ 1 \xrightarrow{b} 6 \in [2] \end{cases}$
- Etat 3 : $\begin{cases} 3 \xrightarrow{a} 6 \in [2] \\ 3 \xrightarrow{b} 5 \in [0] \end{cases}$ Etat 5 : $\begin{cases} 5 \xrightarrow{a} 7 \in [2] \\ 5 \xrightarrow{b} 5 \in [0] \end{cases}$

$\Longrightarrow$ partition de cette classe en 2 sous-classes : $\{0, 1\}$ et $\{3, 5\}$.

Algorithme de Moore-Nerode : Exemple

$\downarrow$ (état initial 0)

Σ	0	1	2	3	4	5	6	7
a	2	7	3	6	5	7	1	1
b	4	6	4	5	2	5	6	6

E

b) Regardons les états de la classe [2] :

- Etat 2 : $\begin{cases} 2 \xrightarrow{a} 3 \in [0] \\ 2 \xrightarrow{b} 4 \in [2] \end{cases}$ Etat 4 : $\begin{cases} 4 \xrightarrow{a} 5 \in [0] \\ 4 \xrightarrow{b} 2 \in [2] \end{cases}$
- Etat 6 : $\begin{cases} 6 \xrightarrow{a} 1 \in [0] \\ 6 \xrightarrow{b} 6 \in [2] \end{cases}$ Etat 7 : $\begin{cases} 7 \xrightarrow{a} 1 \in [0] \\ 7 \xrightarrow{b} 6 \in [2] \end{cases}$

$\Longrightarrow$ pas de partition de cette classe.

Nouvelle partition $E/\sim_1 = \Big\{[0] = \{0,\ 1\}\ ,\ [3] = \{3,\ 5\}\ ,[2] = \{2,\ 4,\ 6,\ 7\}\Big\}$

Algorithme de Moore-Nerode : Exemple

	↓ 0	1	2	3	4	5	6	7	E
a	2	7	3	6	5	7	1	1	
b	4	6	4	5	2	5	6	6	
Σ									

Etape 2 : Construction de $\sim_2$

A partir de $E/\sim_1 = \Big\{[0] = \{0,\ 1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4,\ 6,\ 7\}\Big\}$

Regardons chaque classe d'équivalence séparément :

a) Regardons les états de la classe [0] :

- Etat 0 : $\begin{cases} 0 & \xrightarrow{a} & 2 \in [2] \\ 0 & \xrightarrow{b} & 4 \in [2] \end{cases}$ Etat 1 : $\begin{cases} 1 & \xrightarrow{a} & 7 \in [2] \\ 1 & \xrightarrow{b} & 6 \in [2] \end{cases}$

$\Longrightarrow$ Pas de partition de cette classe

Algorithme de Moore-Nerode : Exemple

$\downarrow$									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	E
a	2	7	3	6	5	7	1	1	
b	4	6	4	5	2	5	6	6	
Σ									

$$E/\sim_1 = \left\{ [0] = \{0,\ 1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4,\ 6,\ 7\} \right\}$$

b) Regardons les états de la classe [3] :

- Etat 3 : $\begin{cases} 3 & \xrightarrow{a} & 6 \in [2] \\ 3 & \xrightarrow{b} & 5 \in [3] \end{cases}$ Etat 5 : $\begin{cases} 5 & \xrightarrow{a} & 7 \in [2] \\ 5 & \xrightarrow{b} & 5 \in [3] \end{cases}$

$\Longrightarrow$ Pas de partition de cette classe

Algorithme de Moore-Nerode : Exemple

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	***E***
a	2	7	3	6	5	7	1	1	
b	4	6	4	5	2	5	6	6	
Σ									

$$E/\sim_1 = \Big\{[0] = \{0,\ 1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4,\ 6,\ 7\}\Big\}$$

c) Regardons les états de la classe [2] :

- Etat 2 : $\begin{cases} 2 & \xrightarrow{a} & 3 \in [3] \\ 2 & \xrightarrow{b} & 4 \in [2] \end{cases}$ Etat 4 : $\begin{cases} 4 & \xrightarrow{a} & 5 \in [3] \\ 4 & \xrightarrow{b} & 2 \in [2] \end{cases}$
- Etat 6 : $\begin{cases} 6 & \xrightarrow{a} & 1 \in [0] \\ 6 & \xrightarrow{b} & 6 \in [2] \end{cases}$ Etat 7 : $\begin{cases} 7 & \xrightarrow{a} & 1 \in [0] \\ 7 & \xrightarrow{b} & 6 \in [2] \end{cases}$

$\Longrightarrow$ partition de cette classe en 2 sous-classes : $\{2,\ 4\}$ et $\{6,\ 7\}$.

Nouvelle partition $E/\sim_2 = \Big\{[0] = \{0,\ 1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4\}\ ,\ [6] = \{6,\ 7\}\Big\}$

Algorithme de Moore-Nerode : Exemple

	↓ 0	1	2	3	4	5	6	7	*E*
a	2	7	3	6	5	7	1	1	
b	4	6	4	5	2	5	6	6	
Σ									

Etape 3 : Construction de $\sim_3$

A partir de $E/\sim_2 = \Big\{[0] = \{0,\ 1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4\}\ ,\ [6] = \{6,\ 7\}\Big\}$

Regardons chaque classe d'équivalence séparément :

a) Regardons les états de la classe [0] :

- Etat 0 : $\left\{\begin{array}{lcl} 0 & \xrightarrow{a} & 2 \in [2] \\ 0 & \xrightarrow{b} & 4 \in [2] \end{array}\right.$ Etat 1 : $\left\{\begin{array}{lcl} 1 & \xrightarrow{a} & 7 \in [6] \\ 1 & \xrightarrow{b} & 6 \in [6] \end{array}\right.$

$\Longrightarrow$ partition de cette classe en 2 sous-classes : $\{0\}$ et $\{1\}$.

Algorithme de Moore-Nerode : Exemple

Σ \ E	↓ **0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
a	2	7	3	6	5	7	1	1
b	4	6	4	5	2	5	6	6

$$E/\sim_2 = \left\{ [0] = \{0,\ 1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4\}\ ,\ [6] = \{6,\ 7\} \right\}$$

b) Regardons les états de la classe [3] :

- Etat 3 : $\begin{cases} 3 & \xrightarrow{a} & 6 \in [6] \\ 3 & \xrightarrow{b} & 5 \in [3] \end{cases}$ Etat 5 : $\begin{cases} 5 & \xrightarrow{a} & 7 \in [6] \\ 5 & \xrightarrow{b} & 5 \in [3] \end{cases}$

$\Longrightarrow$ Pas de partition de cette classe

Algorithme de Moore-Nerode : Exemple

	↓ **0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	***E***
a	2	7	3	6	5	7	1	1	
b	4	6	4	5	2	5	6	6	
Σ									

$$E/\sim_2 = \Big\{[0] = \{0,\ 1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4\}\ ,\ [6] = \{6,\ 7\}\Big\}$$

c) Regardons les états de la classe [2] :

- Etat 2 : $\begin{cases} 2 & \xrightarrow{a} & 3 \in [3] \\ 2 & \xrightarrow{b} & 4 \in [2] \end{cases}$ Etat 4 : $\begin{cases} 4 & \xrightarrow{a} & 5 \in [3] \\ 4 & \xrightarrow{b} & 2 \in [2] \end{cases}$

$\Longrightarrow$ Pas de partition de cette classe

Algorithme de Moore-Nerode : Exemple

$\downarrow$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	***E***
a	2	7	3	6	5	7	1	1	
b	4	6	4	5	2	5	6	6	
$\boldsymbol{\Sigma}$									

$$E/\sim_2 = \Big\{[0] = \{0,\ 1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4\}\ ,\ [6] = \{6,\ 7\}\Big\}$$

d) Regardons les états de la classe [6] :

- Etat 6 : $\left\{\begin{array}{lcl} 6 & \xrightarrow{a} & 1 \in [0] \\ 6 & \xrightarrow{b} & 6 \in [6] \end{array}\right.$　Etat 7 : $\left\{\begin{array}{lcl} 7 & \xrightarrow{a} & 1 \in [0] \\ 7 & \xrightarrow{b} & 6 \in [6] \end{array}\right.$

$\Longrightarrow$ Pas de partition de cette classe

Nouvelle partition

$$E/\sim_3 = \Big\{[0] = \{0\}\ ,\ [1] = \{1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4\}\ ,\ [6] = \{6,\ 7\}\Big\}$$

Algorithme de Moore-Nerode : Exemple

Σ \ E	↓ **0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
a	2	7	3	6	5	7	1	1
b	4	6	4	5	2	5	6	6

Etape 4 : Construction de $\sim_4$

A partir de

$$E/\sim_3 = \Big\{ [0] = \{0\}\ ,\ [1] = \{1\}\ ,\ [3] = \{3, 5\}\ , [2] = \{2, 4\}\ ,\ [6] = \{6, 7\} \Big\}$$

Regardons chaque classe d'équivalence séparément :

a) Les classes [0] et[1] n'ont qu'un élément et ne peuvent donc plus être partitionnées.

Algorithme de Moore-Nerode : Exemple

Σ \ E	$\downarrow$ **0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
a	2	7	3	6	5	7	1	1
b	4	6	4	5	2	5	6	6

$$E/\sim_3 = \Big\{[0] = \{0\}\ ,\ [1] = \{1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4\}\ ,\ [6] = \{6,\ 7\}\Big\}$$

b) Regardons les états de la classe [3] :

- Etat 3 : $\begin{cases} 3 & \xrightarrow{a} & 6 \in [6] \\ 3 & \xrightarrow{b} & 5 \in [3] \end{cases}$ Etat 5 : $\begin{cases} 5 & \xrightarrow{a} & 7 \in [6] \\ 5 & \xrightarrow{b} & 5 \in [3] \end{cases}$

$\Longrightarrow$ Pas de partition de cette classe

Algorithme de Moore-Nerode : Exemple

Σ \ E	→**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
a	2	7	3	6	5	7	1	1
b	4	6	4	5	2	5	6	6

$$E/\sim_3 = \Big\{ [0] = \{0\}\ ,\ [1] = \{1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4\}\ ,\ [6] = \{6,\ 7\} \Big\}$$

c) Regardons les états de la classe [2] :

- Etat 2 : $\begin{cases} 2 & \xrightarrow{a} & 3 \in [3] \\ 2 & \xrightarrow{b} & 4 \in [2] \end{cases}$ Etat 4 : $\begin{cases} 4 & \xrightarrow{a} & 5 \in [3] \\ 4 & \xrightarrow{b} & 2 \in [2] \end{cases}$

$\Longrightarrow$ Pas de partition de cette classe

Algorithme de Moore-Nerode : Exemple

$\downarrow$

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	***E***
a	2	7	3	6	5	7	1	1	
b	4	6	4	5	2	5	6	6	
Σ									

$$E/\sim_3 = \Big\{[0] = \{0\}\ ,\ [1] = \{1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4\}\ ,\ [6] = \{6,\ 7\}\Big\}$$

d) Regardons les états de la classe [6] :

- Etat 6 : $\begin{cases} 6 & \xrightarrow{a} & 1 \in [1] \\ 6 & \xrightarrow{b} & 6 \in [6] \end{cases}$ Etat 7 : $\begin{cases} 7 & \xrightarrow{a} & 1 \in [1] \\ 7 & \xrightarrow{b} & 6 \in [6] \end{cases}$

$\Longrightarrow$ Pas de partition de cette classe

$$E/\sim_4 = E/\sim_3 = \Big\{[0] = \{0\}\ ,\ [1] = \{1\}\ ,\ [3] = \{3,\ 5\}\ , [2] = \{2,\ 4\}\ ,\ [6] = \{6,\ 7\}\Big\}$$

Algorithme de Moore-Nerode : Exemple

$$E/\sim_4 = E/\sim_3 = \Big\{[0] = \{0\}\,,\ [1] = \{1\}\,,\ [3] = \{3,\,5\}\,,[2] = \{2,\,4\}\,,\ [6] = \{6,\,7\}\Big\}$$

$\Longrightarrow$ $\sim_4 = \sim_3$

$\Longrightarrow$ on s'arrête

États à fusionner :

- état fusionné $\{3,\ 5\}$ non acceptant comme 1 et 5
- état fusionné $\{2,\ 4\}$ acceptant car 2 et 4 le sont
- état fusionné $\{6,\ 7\}$ acceptant car 6 et 7 le sont

On obtient l'automate moore optimisé ci-dessous

Automate de départ	Automate optimisé