LOG 2810: Structures discrètes

Module : Théorie des Langages

Julien Dompierre, Philippe Galinier, Robert Roy
Mis à jour par
John Mullins, Fayçal Abouzaid et Foutse Khomh

Ecole Polytechnique de Montréal

Liste des Chapitres

Grammaires

Langages et grammaires

Définition: le langage naturel

La **syntaxe**: règles qui président à l'ordre des mots et à la construction des phrases dans une langue.

La **sémantique**: étude du langage considéré du point de vue du sens.

Un **langage naturel** est une langue comme le français ou le tagalog.

La syntaxe d'un langage naturel est extrêmement complexe et en fait il ne semble pas possible de préciser toutes les règles syntaxiques.

Définition: le langage formel

Le **langage formel**, contrairement au langage naturel, est défini par un ensemble spécifique de règles de syntaxe.

La **grammaire** est une liste de règles de syntaxe qui permet, d'une part, de déterminer si une combinaison de mots constitue une phrase valide, et d'autre part, de créer des phrases valides dans un langage formel.

Définitions de base

Un **vocabulaire** (ou **alphabet**) V est un ensemble fini non vide d'éléments appelés des **symboles**.

Un **mot** dans V est une chaîne de longueur finie d'éléments de V. La **longueur** $l(w)$ d'un mot w est le nombre d'éléments de V que contient le mot.

La **concaténation** des mots w_0 et w_1 est le mot w formé des éléments de w_0 suivit des éléments du w_1.

La **chaîne vide**, notée ϵ, est la chaîne qui ne contient aucun symbole.

L'ensemble de tous les mots dans V est noté V^*.

Un **langage dans** V est un sous-ensemble de V^*.

Grammaire syntagmatique

Une **grammaire syntagmatique** $G = (V, T, S, P)$ est constituée d'un vocabulaire V, d'un sous-ensemble T de symboles terminaux, d'un symbole de départ S de V (axiome) et d'un ensemble P de règles de $(V^* \rightarrow V^*)$ de production.

L'ensemble $V - T$ est noté N. Les éléments de N sont appelés les **symboles non terminaux**.

Une production est notée $w_0 \rightarrow w_1$, où $w_0, w_1 \in V^*$ et w_0 contient au moins un non terminal.

Définition: mots dérivables

Soit $G = (V, T, S, P)$ une grammaire syntagmatique. Soit $w_0 = lz_0r$ et $w_1 = lz_1r$ des chaînes dans V. Si $z_0 \to z_1$ est une production de G, on dit que w_1 est **directement dérivable** à partir de w_0 et on écrit $w_0 \Rightarrow w_1$.

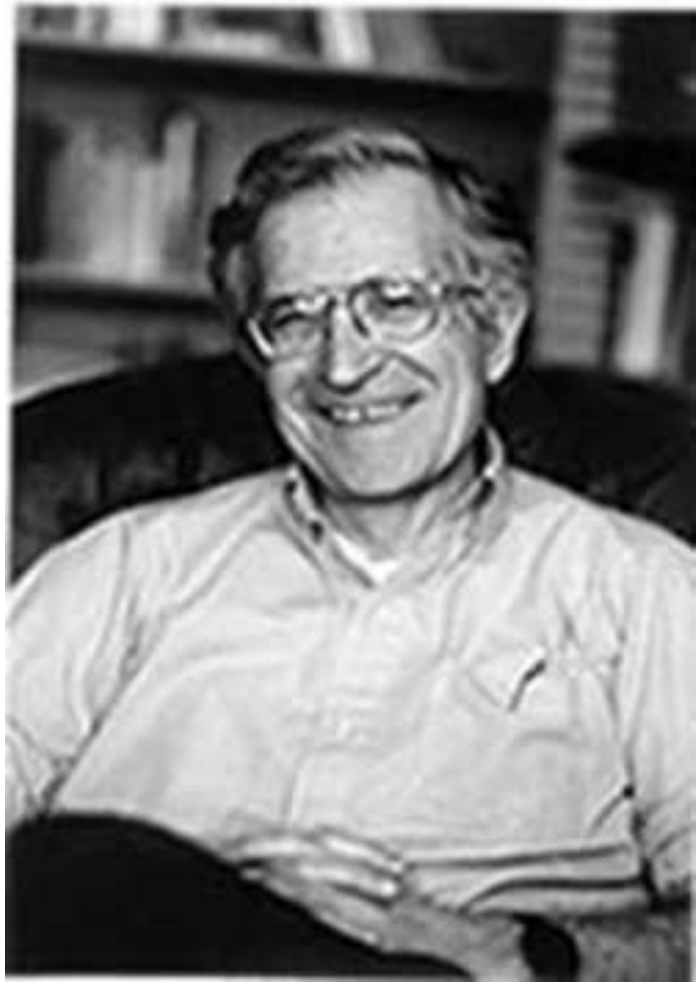
Si $w_0, w_1, ..., w_n$ où $n \geq 1$, sont des chaînes dans V telles que $w_0 \Rightarrow w_1$, $w_1 \Rightarrow w_2$, ..., $w_{n-1} \Rightarrow w_n$, alors on dit que w_n **est dérivable à partir de** w_0 et on écrit $w_0 \overset{*}{\Rightarrow} w_n$. La suite des étapes utilisées pour obtenir w_n à partir de w_0 s'appelle une **dérivation**.

Définition: langage de G

Soit $G = (V, T, S, P)$ une grammaire syntagmatique. Le **langage créé à partir de** G (ou le **langage de** G), noté $L(G)$, est l'ensemble de toutes les chaînes de terminaux qui sont dérivables à partir de l'axiome S. En d'autres mots,

$$L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}.$$

Note historique: Noam Chomsky



Né le 7 décembre 1928 à Philadelphie.

MIT Linguistics and Philosophy, 77 Massachusetts Avenue Bldg. 32-D808 Cambridge, MA 02139 USA.
chomsky@mit.edu
`web.mit.edu/linguistics/www/chomsky.home.html`

Hiérarchie de Chomsky

Chomsky a réalisé une une classification naturelle des langages décrits par les grammaires formelles, dite “**La hiérarchie de Chomsky**”.

1. grammaire de type 0 : Grammaires généralisées
2. grammaire de type 1 : Grammaires contextuelles
3. grammaire de type 2 : Grammaires hors-contexte
4. grammaire de type 3 : Grammaires régulières

Définition: grammaire de type 0

Une grammaire de **type 0** n'a aucune restriction quant à ses productions.

Les grammaires de type 0 sont appelées des **grammaires récursivement énumérables** ou **grammaires généralisées**

Définition: grammaire de type 1

Une grammaire de **type 1** ne peut que comporter des productions ayant la forme $w_1 \rightarrow w_2$, où la longueur de w_2 est plus grande que ou égale à la longueur de w_1, ou de la forme $w_1 \rightarrow \epsilon$.

Lorsqu'il y a production de la forme $lw_1r \rightarrow lw_2r$ (mais pas de la forme $w_1 \rightarrow w_2$, la grammaire est de type 1, ou **contextuelle** puisqu'on peut remplacer w_1 par w_2 uniquement lorsqu'il est entouré des chaînes l et r.

Définition: grammaire de type 2

Une grammaire de **type 2** peut avoir des productions ayant la forme $w_1 \rightarrow w_2$ seulement, où w_1 est un symbole unique qui n'est pas un symbole terminal.

Les grammaires de type 2 sont appelées des **grammaires algébriques** ou **grammaires hors contexte** (GHC) (ou non contextuelle, ou libre de contexte) puisqu'on peut remplacer un symbole non terminal, qui est le côté gauche de la production, dans une chaîne lorsqu'il apparaît, peu importe ce qui se trouve dans la chaîne.

Un langage créé par une grammaire de type 2 est appelé un **langage algébrique**.

Définition: grammaire de type 3

Une grammaire de **type 3** peut uniquement comprendre des productions ayant la forme $w_1 \rightarrow w_2$ avec $w_1 = A$ et $w_2 = aB$ ou $w_2 = a$, où A et B sont des symboles non terminaux et a est un symbole terminal, ou avec $w_1 = S$ et $w_2 = \epsilon$.

Les grammaires de type 3 sont appelées des **grammaires régulières**. Les langages produits par une grammaire régulière sont **réguliers**.

Types de grammaires

Type	Restrictions sur les productions $w_1 \rightarrow w_2$
0	Aucune restriction
1	$l(w_1) \leq l(w_2)$ ou bien $w_1 \rightarrow \epsilon$.
2	$w_1 = A$, où A est un symbole non terminal
3	$w_1 = A$ et $w_2 = aB$ ou $w_2 = a$, où $A \in N$, $B \in N$ et $a \in T$ ou $S \rightarrow \epsilon$

Toute grammaire de type 3 est une grammaire de type 2, toute grammaire de type 2 est une grammaire de type 1 et toute grammaire de type 1 est une grammaire de type 0.

Module

Machines à états finis avec sortie

Modélisation d'un distributeur automatique

Le prix d'une boisson est de 30 cents. On peut mettre des pièces de 5 cents, 10 cents ou 25 cents. Si on met plus de 30 cents, la machine garde 30 cents et rend la monnaie. On peut appuyer sur le bouton 1 pour obtenir un Coca-Cola ou sur la bouton 2 pour obtenir un Pepsi. La machine ne plante pas et ne peut pas être rebootée!

Modélisation d'un distributeur automatique

L'ensemble des entrées possibles I de la machine sont $5¢$, $10¢$, $25¢$, bouton $B1$ et bouton $B2$.

L'ensemble des sorties possibles O de la machine sont rien (n), $5¢$, $10¢$, $15¢$, $20¢$, $25¢$, un Coca-Cola C ou un Pepsi P.

La machine doit conserver en mémoire la somme d'argent déposée jusqu'à maintenant. Les états possibles S de sa mémoire sont $0¢$, $5¢$, $10¢$, $15¢$, $20¢$, $25¢$ et $30¢$. Ces états sont notés s_0, s_1, ..., s_6 respectivement.

Fonction de transition pour un distributeur

Dans quel état se trouve la machine en fonction de l'entrée.

État	État suivant				
	Entrée				
	5	10	25	$B1$	$B2$
s_0 $(0¢)$	s_1	s_2	s_5	s_0	s_0
s_1 $(5¢)$	s_2	s_3	s_6	s_1	s_1
s_2 $(10¢)$	s_3	s_4	s_6	s_2	s_2
s_3 $(15¢)$	s_4	s_5	s_6	s_3	s_3
s_4 $(20¢)$	s_5	s_6	s_6	s_4	s_4
s_5 $(25¢)$	s_6	s_6	s_6	s_5	s_5
s_6 $(30¢)$	s_6	s_6	s_6	s_0	s_0

Fonction de sortie pour un distributeur

Que retourne la machine en fonction de l'entrée.

État	Sortie				
	Entrée				
	5	10	25	$B1$	$B2$
s_0 (0¢)	n	n	n	n	n
s_1 (5¢)	n	n	n	n	n
s_2 (10¢)	n	n	5	n	n
s_3 (15¢)	n	n	10	n	n
s_4 (20¢)	n	n	15	n	n
s_5 (25¢)	n	5	20	n	n
s_6 (30¢)	5	10	25	C	P

Note historique: George H. Mealy

Il travaillait chez Bell Labs et a publié “*A Method for Synthesizing Sequential Circuits*”, Bell System Tech. J. vol 34, pp. 1045 à1079, septembre 1955.

Définition: machine à états finis avec sorties

Une **machine à états finis** $M = (S, I, O, f, g, s_0)$ est constituée

- d'un ensemble fini d'**états** S,
- d'un **alphabet d'entrée** fini I,
- d'un **alphabet de sortie** fini O,
- d'une **fonction de transition** $f : S \times I \to S$ qui attribue un nouvel état à chaque couple (état, entrée),
- d'une **fonction de sortie** $g : S \times I \to O$ qui attribue une sortie à chaque couple (état, entrée),
- d'un **état initial** s_0.
- Ces machines sont appelées machines de Mealy.

Modélisation d'un distributeur automatique

Un distributeur de boissons gazeuses est une machine à états finis $M = (S, I, O, f, g, s_0)$ où

- $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$,
- $I = \{5, 10, 25, B1, B2\}$,
- $O = \{n, 5, 10, 15, 20, 25, C, P\}$,
- l'état initial est s_0,
- suite, les deux transparents suivants...

Modélisation d'un distributeur automatique (suite)

Un distributeur de boissons gazeuses est une machine à états finis $M = (S, I, O, f, g, s_0)$ où

- $f : S \times I \to S$ est une fonction qui attribue un nouvel état à chaque couple (état, entrée),
- suite, transparent suivant...

S	$f : S \times I \to S$				
	I				
	5	10	25	$B1$	$B2$
s_0	s_1	s_2	s_5	s_0	s_0
s_1	s_2	s_3	s_6	s_1	s_1
s_2	s_3	s_4	s_6	s_2	s_2
s_3	s_4	s_5	s_6	s_3	s_3
s_4	s_5	s_6	s_6	s_4	s_4
s_5	s_6	s_6	s_6	s_5	s_5
s_6	s_6	s_6	s_6	s_0	s_0

Modélisation d'un distributeur automatique (fin)

Un distributeur de boissons gazeuses est une machine à états finis $M = (S, I, O, f, g, s_0)$ où

- $g : S \times I \to O$ est une fonction qui attribue une sortie à chaque couple (état, entrée).

S	$g : S \times I \to O$				
	I				
	5	10	25	$B1$	$B2$
s_0	n	n	n	n	n
s_1	n	n	n	n	n
s_2	n	n	5	n	n
s_3	n	n	10	n	n
s_4	n	n	15	n	n
s_5	n	5	20	n	n
s_6	5	10	25	C	P

Définition: table d'états

Soit $M = (S, I, O, f, g, s_0)$ une machine à états finis. On peut utiliser une **table d'états** (ou **table de transition d'états**) pour représenter les valeurs de la fonction de transition f et de la fonction de sortie g pour tous les couples d'états et d'entrée.

Table d'états pour le distributeur automatique

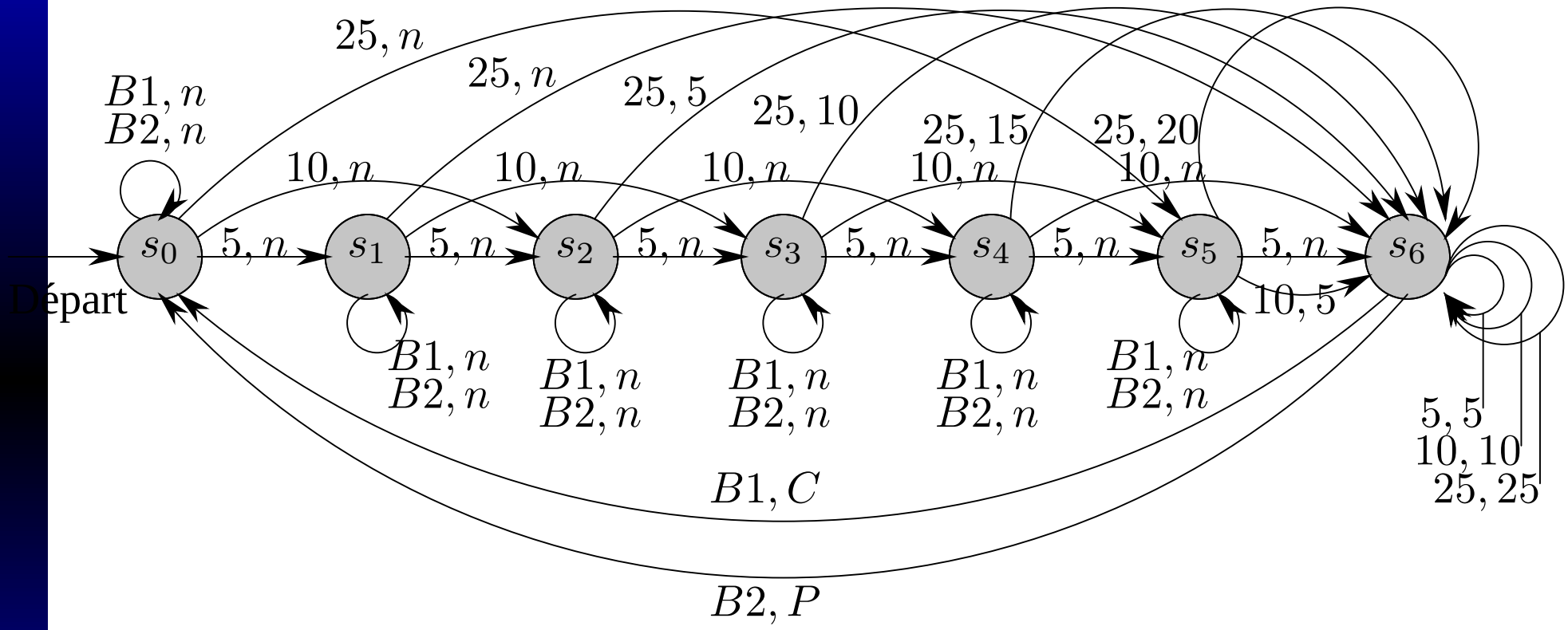
S	$f : S \times I \to S$					$g : S \times I \to O$				
	I					I				
	5	10	25	$B1$	$B2$	5	10	25	$B1$	$B2$
s_0	s_1	s_2	s_5	s_0	s_0	n	n	n	n	n
s_1	s_2	s_3	s_6	s_1	s_1	n	n	n	n	n
s_2	s_3	s_4	s_6	s_2	s_2	n	n	5	n	n
s_3	s_4	s_5	s_6	s_3	s_3	n	n	10	n	n
s_4	s_5	s_6	s_6	s_4	s_4	n	n	15	n	n
s_5	s_6	s_6	s_6	s_5	s_5	n	5	20	n	n
s_6	s_6	s_6	s_6	s_0	s_0	5	10	25	C	P

Définition: diagramme d'états

Soit $M = (S, I, O, f, g, s_0)$ une machine à états finis. On peut la représenter en utilisant un **diagramme d'états**, qui est un graphe orienté avec arcs étiquetés. Dans de diagramme,

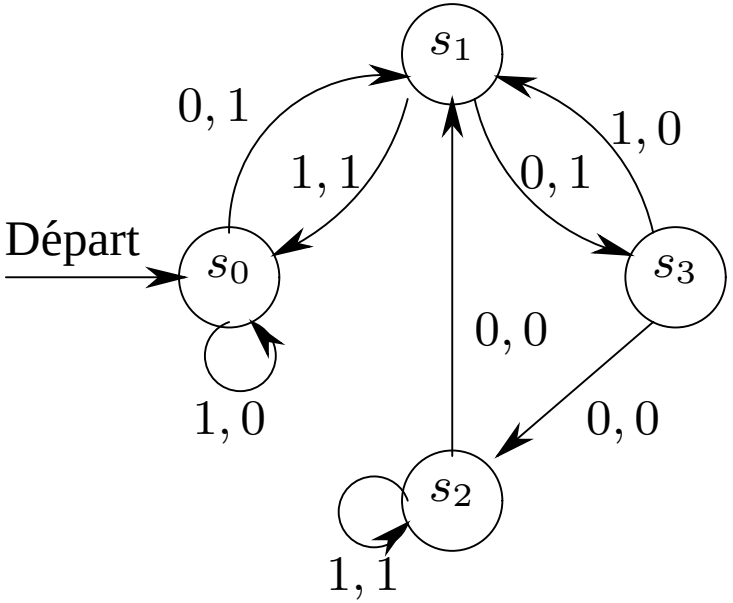
- chaque état est représenté par un cercle.
- chaque transition entre deux états est représentée par un arc orienté et étiqueté,
- chaque étiquette est une paire (valeur d'entrée, valeur de sortie).

Diagramme d'états pour le distributeur automatique



Exemple table d'états – diagramme d'états

S	f		g	
	I		I	
	0	1	0	1
s_0	s_1	s_0	1	0
s_1	s_3	s_0	1	1
s_2	s_1	s_2	0	1
s_3	s_2	s_1	0	0



Chaîne de sortie produite par une machine

Une chaîne d'entrée fait passer l'état de départ par une suite d'états qui sont déterminés par la fonction de transition. A la lecture de la chaîne d'entrée, symbole par symbole de gauche à droite, chaque symbole d'entrée fait passer la machine d'un état à un autre, puisque chaque transition produit une sortie, toute chaîne d'entrée produit une chaîne de sortie.

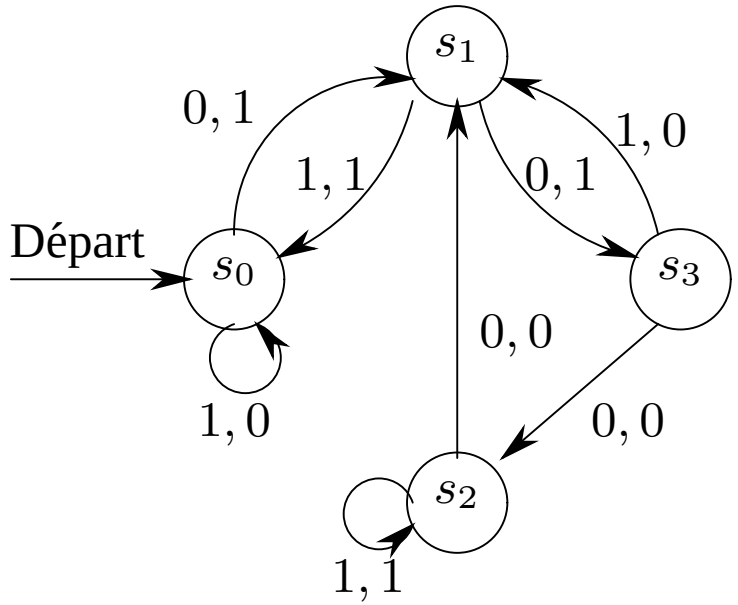
Chaîne de sortie produite par une machine

On suppose que la chaîne d'entrée est $x = x_1 x_2 \cdots x_k$. Alors, la lecture de cette entrée fait passer la machine de l'état s_0 à l'état s_1 où $s_1 = f(s_0, x_1)$, puis à l'état s_2 où $s_2 = f(s_1, x_2)$, et ainsi de suite, ce parcours se terminant à l'état s_k où $s_k = f(s_{k-1}, x_k)$. Cette suite de transitions produit la chaîne de sortie $y = y_1 y_2 \cdots y_k$ où $y_1 = g(s_0, x_1)$ est la sortie correspondant à la transition de s_0 à s_1, $y_2 = g(s_1, x_2)$ est la sortie correspondant à la transition de s_1 à s_2 et ainsi de suite.

On peut étendre la définition de la fonction de sortie g pour qu'elle s'applique aux chaînes d'entrée de telle sorte que $g(x) = y$ où y est la chaîne produite correspondant à la chaîne d'entrée x.

Exemple de sortie produite par un machine

S	f		g	
	I		I	
	0	1	0	1
s_0	s_1	s_0	1	0
s_1	s_3	s_0	1	1
s_2	s_1	s_2	0	1
s_3	s_2	s_1	0	0

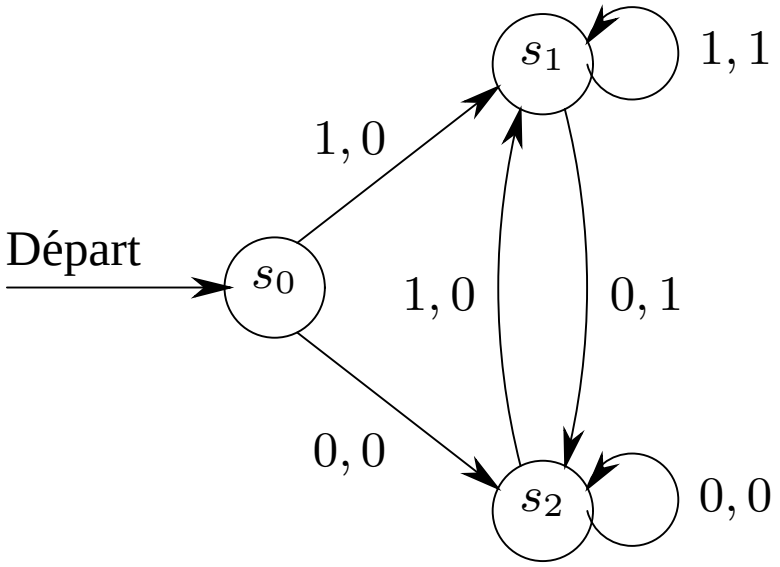


I	1	1	0	1	1	0	1	1
S	s_0	s_0	s_0	s_1	s_0	s_0	s_1	s_0	s_0
O	0	0	1	1	0	1	1	0	

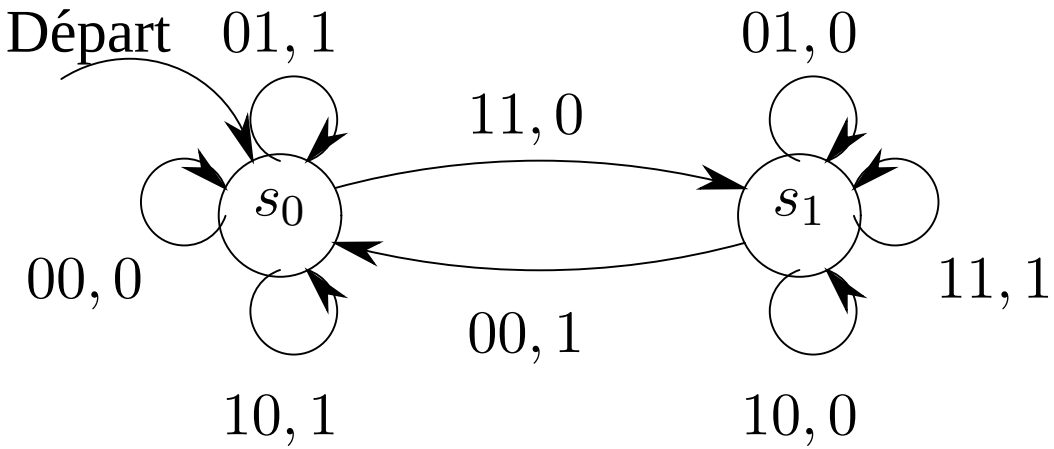
Exemple de machine de Mealy: le retardateur

Le retardateur transforme la chaîne binaire $x_1x_2\cdots x_k$ en la chaîne $0x_1x_2\cdots x_{k-1}$. La machine est à l'état s_1 si l'entrée précédente était 1 et à l'état s_2 si l'entrée précédente était 0.

S	f		g	
	I		I	
	0	1	0	1
s_0	s_2	s_1	0	0
s_1	s_2	s_1	1	1
s_2	s_2	s_1	0	0



Exemple de machine de Mealy: addition binaire



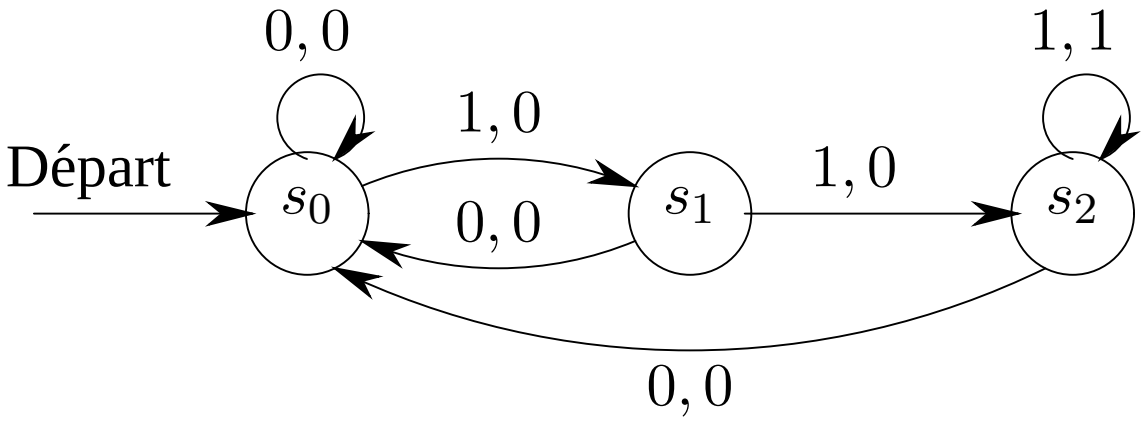
S	f				g			
	I				I			
	00	01	10	11	00	01	10	11
s_0	s_0	s_0	s_0	s_1	0	1	1	0
s_1	s_0	s_1	s_1	s_1	1	0	0	1

Exemple de machine qui reconnaît un langage

On veut une machine à états finis qui produit un 1 comme bit de sortie si et seulement si les trois derniers bits reçus sont tous des 1. Cette machine reconnaît un langage car elle produit 1 si et seulement si la chaîne d'entrée admet une propriété spécifique.

L'état s_0 se souvient que l'entrée précédente n'était pas un 1. L'état s_1 se souvient que l'entrée précédente était un 1, mais l'antépénultième entrée n'était pas un 1. L'état s_1 se souvient que les deux entrées précédentes étaient des 1.

Exemple de machine qui reconnaît un langage



S	f		g	
	I		I	
	0	1	0	1
s_0	s_0	s_1	0	0
s_1	s_0	s_2	0	0
s_2	s_0	s_2	0	1

Note historique: Edward F. Moore

Edward F. Moore né le 23 novembre 1925 à Baltimore au Maryland et mort le 14 juin 2003 à Madison au Wisconsin.

Il travaillait chez Bell Labs et a publié "*Gedanken-experiments on sequential machines*", pp. 129-153 dans Shannon and McCarthy (éds.), Automata Studies, Annals of Mathematics Studies, Number 34, Princeton University Press, 1956.

Définition: machine de Moore

Une **machine de Moore** $M = (S, I, O, f, g, s_0)$ est constituée

- d'un ensemble fini d'états S,
- d'un alphabet d'entrée fini I,
- d'un alphabet de sortie fini O,
- d'une fonction de transition $f : S \times I \to S$ qui attribue un nouvel état à chaque couple (état, entrée),
- d'une fonction de sortie $g : S \to O$ *qui attribue une sortie à chaque état*,
- d'un état initial s_0.

table d'états – diagramme d'états

S	f		g
	I		
	0	1	
s_0	s_0	s_2	0
s_1	s_3	s_0	1
s_2	s_2	s_1	1
s_3	s_2	s_0	1

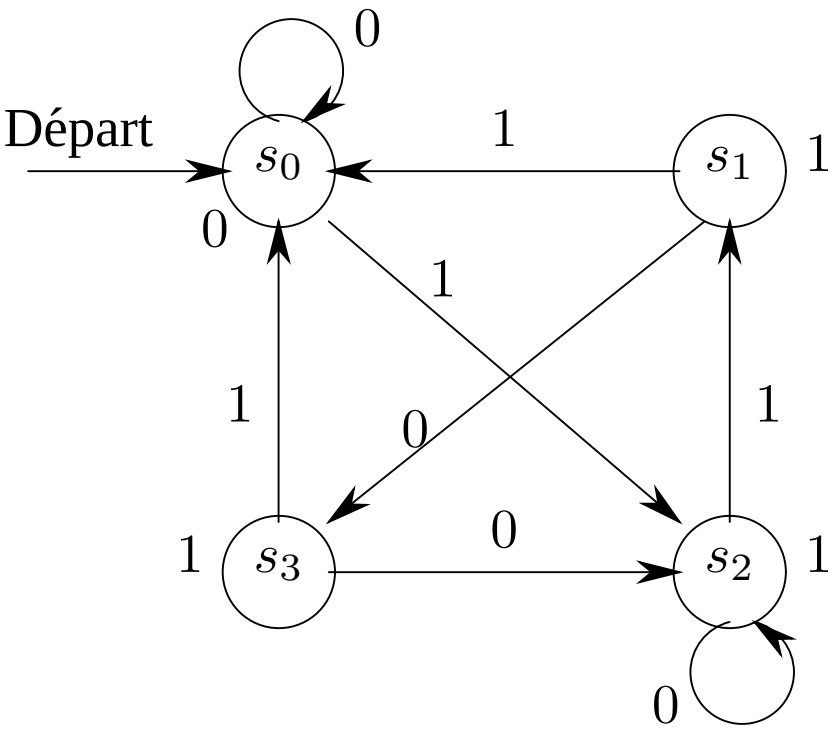
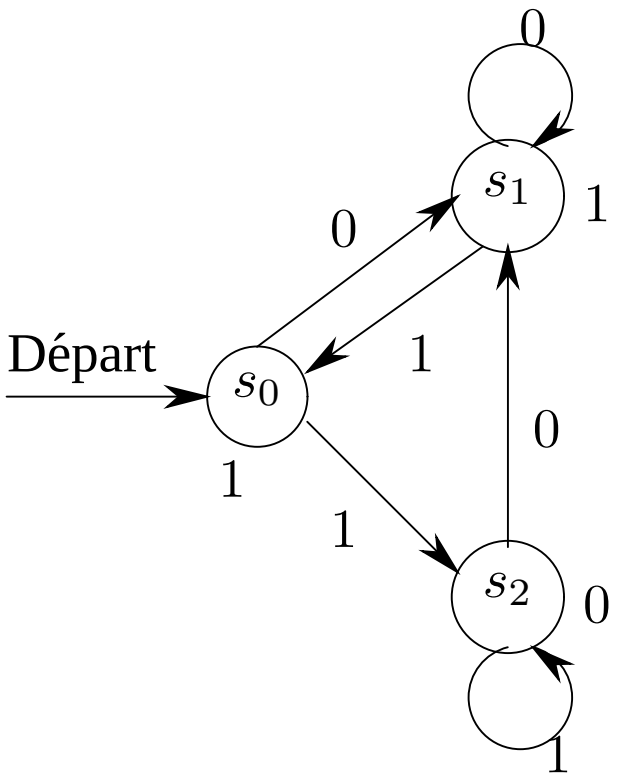


diagramme d'états – table d'états



S	f		g
	I		
	0	1	
s_0	s_1	s_2	1
s_1	s_1	s_0	1
s_2	s_1	s_2	0

Conversion Moore-Mealy

THÉORÈME :
Pour toute machine de Moore $M = (S,\ I,\ O, f,\ g,\ s_0)$, il existe une machine de Mealy $M' = (S',\ I',\ O',\ f',\ g',\ s'_0)$ qui possède les mêmes fonctionnements.

PREUVE :

On pose :

- $I' = I, O' = O, S' = S$ et $s'_0 = s_0$,

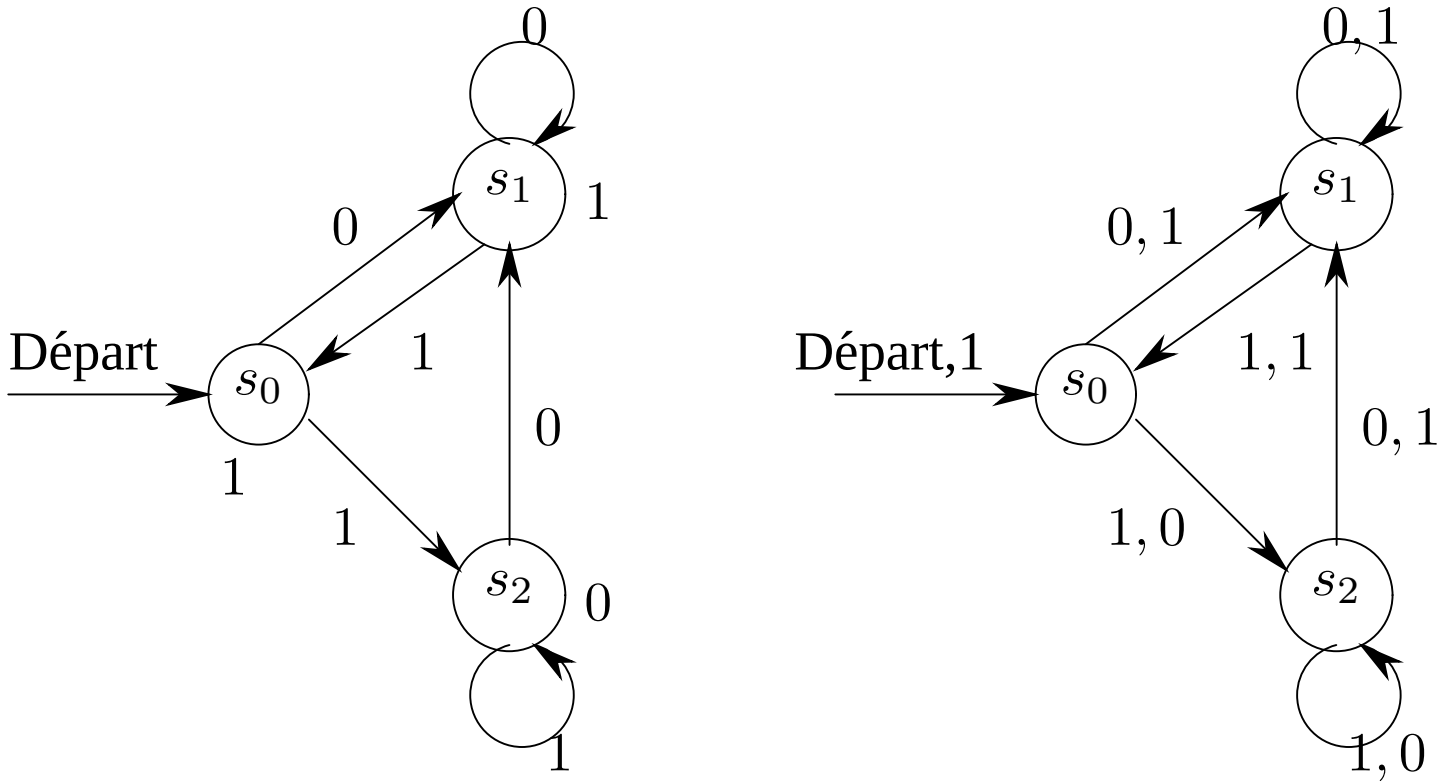
et f' vérifie : $f'(q,\ e) = q'$ et $s = g'(q)$ ssi $f(q,\ e) = q'$ et $s = g(q')$

On prouve que tout fonctionnement de l'une est un fonctionnement de l'autre par récurrence sur la longueur des fonctionnements.

Un fonctionnement est le couple de mots $(w_1 w_2)$ tq $w_1 = e_1 \ldots e_n$ des entrées qui produisent les sorties $w_2 = s_1 \ldots s_n$ passage par les états.

Conversion Moore-Mealy

Conversion d'une machine de Moore, sous forme de diagramme d'états, en une machine de Mealy.



Conversion Moore-Mealy

Conversion d'une machine de Moore, sous forme de table d'états, en une machine de Mealy.

S	f		g
	I		
	0	1	
s_0	s_1	s_2	1
s_1	s_1	s_0	1
s_2	s_1	s_2	0

S	f		g	
	I		I	
	0	1	0	1
s_0	s_1	s_2	1	0
s_1	s_1	s_0	1	1
s_2	s_1	s_2	1	0

Conversion Mealy-Moore

THÉORÈME :
Pour toute machine de Mealy $M = (S,\ I,\ O, f,\ g,\ s_0)$, il existe une machine de Moore $M' = (S',\ I',\ O',\ f',\ g',\ s'_0)$ qui possède les mêmes fonctionnements.

PREUVE :

Si la machine de Mealy a N état, et P sorties, la machine de Moore aura au plus $P \times N + 1$ états .

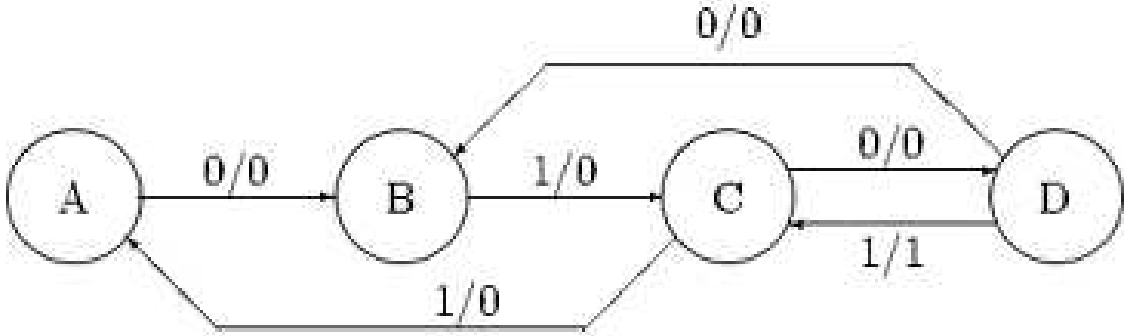
Un état qui reçoit des transitions avec plus d'une sortie est 'splitté' en autant d'états que de sorties. On aura donc:

- $I' = I, O' = O, S' = S \times O$ et $s'_0 = s_0$,
- $f'((q, s), e) = (q', s')$ et $g'(q) = s$ ssi $f(q, e) = q')$ et $g'(q, e) = s'$

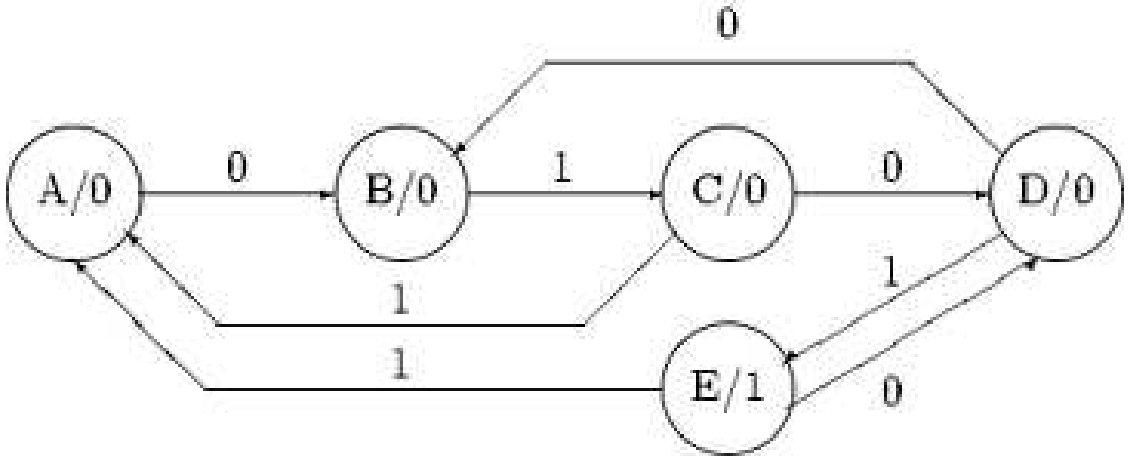
Preuve de l'équivalence par récurrence sur la longueur des fonctionnements.

Exemple de Conversion Mealy-Moore

Machine de Mealy



Machine de Moore équivalente



Automates d'États Finis

Machines à états finis sans sortie

Définition: concaténation et puissance

DÉFINITION: concaténation

Soient A et B des *sous-ensembles* de V^*, où V est un vocabulaire. La **concaténation** de A et B, notée $A.B$, est l'ensemble de toutes les chaînes de la forme xy où x est une chaîne dans A et y est une chaîne dans B.

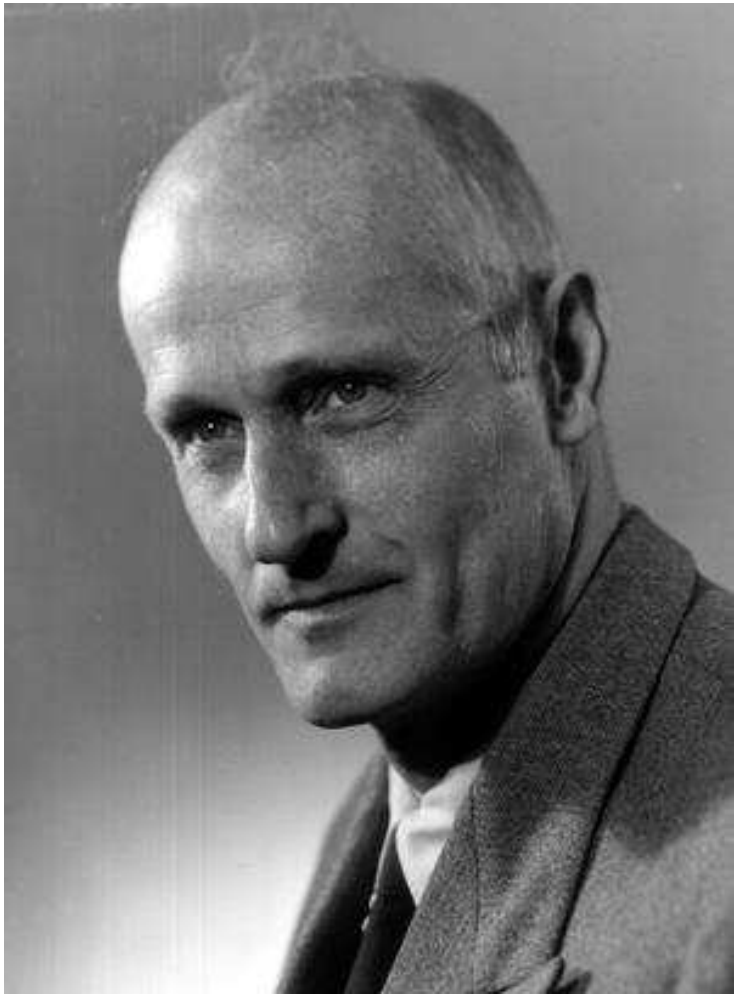
DÉFINITION: puissances de A

On définit récursivement A^n pour $n = 0, 1, 2, ...$

- $A^0 = \{\epsilon\}$
- $A^{n+1} = A^n.A$ pour $n = 1, 2, 3, ...$

Note historique: Stephen Cole Kleene

Né le 5 janvier 1909 à Hartford, Connecticut. Mort le 25 janvier 1994 à Madison, Wisconsin



Kleene prononçait son nom de famille "klay'nee". "klee'nee" et "kleen" sont deux mauvaises prononciations très répandues.

`math.library.wisc.edu/kleene.htm`

Définition: fermeture de Kleene

On suppose que A est un sous-ensemble de V^*. Alors, la **fermeture de Kleene** de A, notée A^*, est l'ensemble constitué des concaténations d'un nombre arbitraire de chaînes de A. Autrement dit,

$$A^* = \bigcup_{k=0}^{\infty} A^k.$$

Définition: machine à états finis sans sortie

Une **machine à états finis sans sortie**, également appelée **automate fini**, $M = (S, I, f, s_0, F)$ est constituée

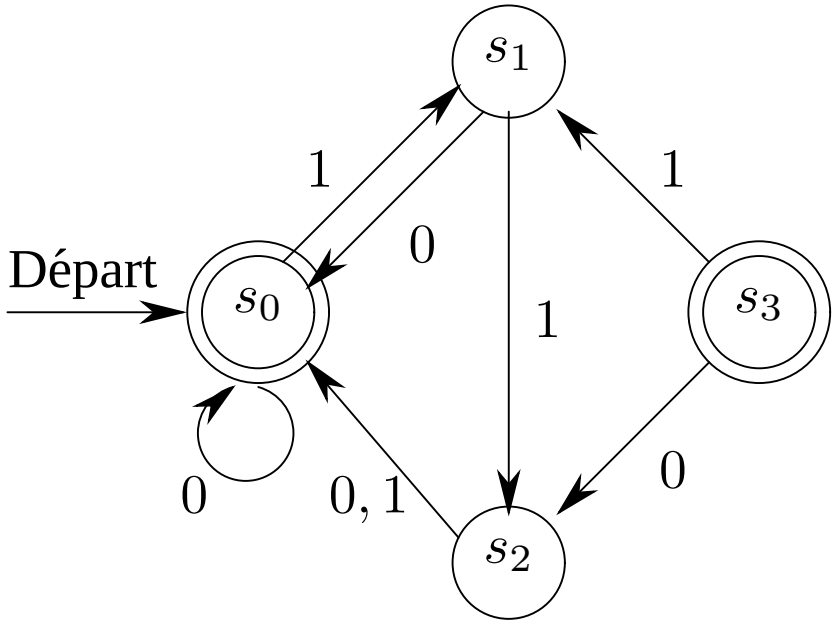
- d'un ensemble fini d'états S,
- d'un alphabet d'entrée fini I,
- d'une fonction de transition $f : S \times I \to S$ qui attribue un état suivant à chaque couple (état, entrée),
- d'un état initial s_0,
- et d'un sous-ensemble F de S constitué d'états finaux.

On peut représenter un automate fini en utilisant soit des tables d'états, soit des diagrammes d'états. Les états finaux sont indiqués dans les diagrammes d'états par des cercles doubles.

Table d'états - Diagramme d'états. Exemple.

Soit l'automate fini $M = (S, I, f, s_0, F)$, où $S = \{s_0, s_1, s_2, s_3\}$, $I = \{0, 1\}$, $F = \{s_0, s_3\}$.

	f	
S	I	
	0	1
s_0	s_0	s_1
s_1	s_0	s_2
s_2	s_0	s_0
s_3	s_2	s_1



Définition: langage reconnu

On dit qu'une chaîne est **reconnue** ou **acceptée** par la machine $M = (S, I, f, s_0, F)$ si elle fait passer l'état initial s_0 à un état final, autrement dit si $f(s_0, x)$ est un état dans F.

Le **langage reconnu** ou **langage accepté** par la machine M, noté $L(M)$, est l'ensemble de toutes les chaînes qui sont reconnues par M.

Deux automates finis sont **équivalents** s'il reconnaissent le même langage.

Automates finis déterministes et non déterministes. Exemples

Un automate fini est **déterministe** ou **à comportement déterminé**, si pour chaque couple (état, entrée), il y a un état suivant unique donné par la fonction de transition f.

Dans un automate fini **non déterministe**, il peut y avoir aucun, un ou plusieurs états suivants donnés par la fonction de transition f.

Définition: automate fini non déterministe

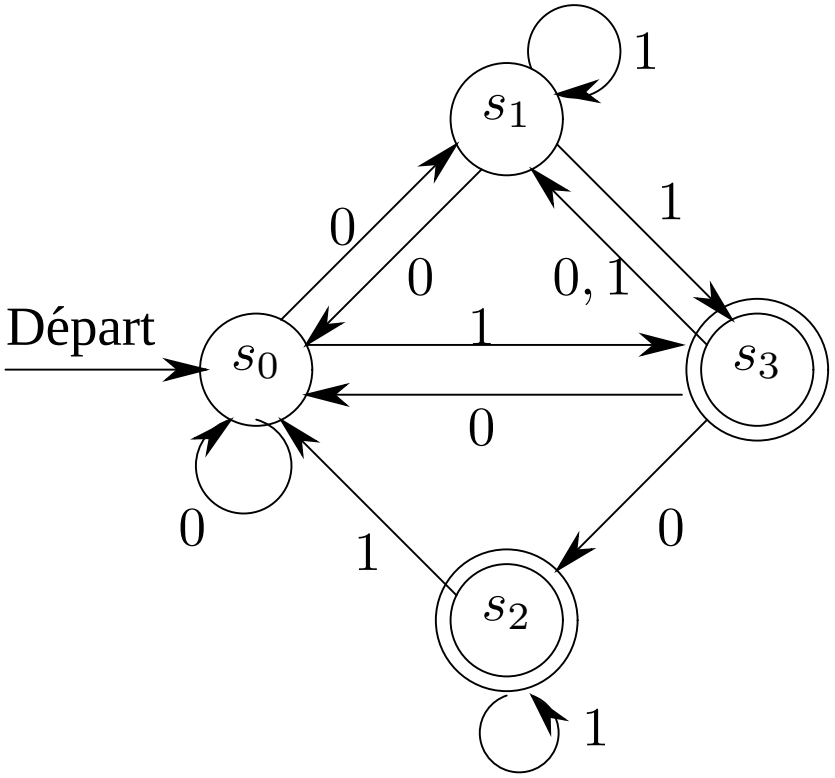
Un **automate fini non déterministe (AFN)** $M = (S, I, f, s_0, F)$ est constitué

- d'un ensemble fini d'états S,
- d'un alphabet d'entrée fini I,
- d'une fonction de transition $f : S \times I \rightarrow \mathcal{P}(S)$ qui attribue un *ensemble d'états* à chaque couple (état, entrée),
- d'un état initial s_0,
- et d'un sous-ensemble F de S constitué d'états finaux.

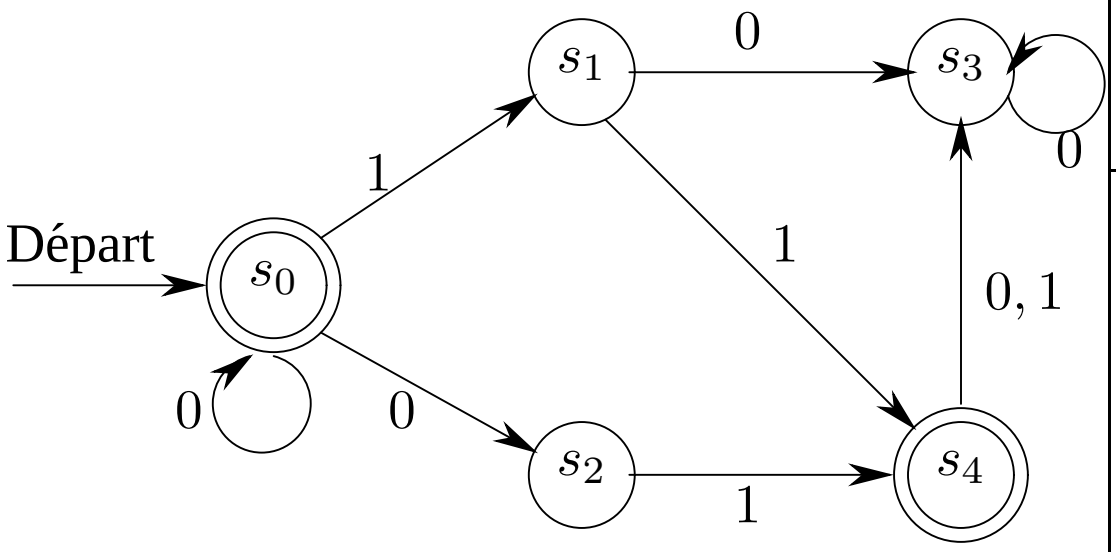
Table et Diagramme d'états pour un AFN

Soit l'automate fini $M = (S, I, f, s_0, F)$, où
$S = \{s_0, s_1, s_2, s_3\}$, $I = \{0, 1\}$, $F = \{s_2, s_3\}$.

	f	
S	I	
	0	1
s_0	$\{s_0, s_1\}$	$\{s_3\}$
s_1	$\{s_0\}$	$\{s_1, s_3\}$
s_2	$\varnothing$	$\{s_0, s_2\}$
s_3	$\{s_0, s_1, s_2\}$	$\{s_1\}$



Exemple diagramme d'états – table d'états



	f	
S	I	
	0	1
s_0	$\{s_0, s_2\}$	$\{s_1\}$
s_1	$\{s_3\}$	$\{s_4\}$
s_2	$\varnothing$	$\{s_4\}$
s_3	$\{s_3\}$	$\varnothing$
s_4	$\{s_3\}$	$\{s_3\}$

Fonction de transition pour une chaîne

Que signifie pour un automate fini non déterministe le fait de reconnaître une chaîne?

Soit $x = x_1 x_2 \cdots x_k$, une chaîne dans I^*. Alors, $f(s_1, x)$ est un *ensemble d'états* obtenu en utilisant chaque symbole successif de x, de gauche à droite, comme entrée en commençant par l'état s_1. À partir de s_1, on passe à un ensemble d'états $S_2 = f(s_1, x_1)$, puis le symbole x_2 fait passer chacun des états de S_2 à un ensemble d'états. Soit S_3 l'union de ces ensembles, ie, $S_3 = f(S_2, x_2)$. On poursuit ce processus, en incluant à chaque étape tous les états obtenus en utilisant un état obtenu à l'étape précédente et le symbole d'entrée actuel.

Définition: langage reconnu

On dit que la chaîne x est **reconnue** ou **acceptée** par l'automate fini non déterministe $M = (S, I, f, s_0, F)$ s'il y a un état final dans l'ensemble de tous les états qu'on peut obtenir à partir de s_0 en consommant x.

Le **langage reconnu** ou **langage accepté** par la machine M, noté $L(M)$, est l'ensemble de toutes les chaînes qui sont reconnues par M.

Déterminisation d'un AFN

THÉORÈME: Si le langage L est reconnu par un automate fini *non déterministe* M_0, alors il existe un automate fini *déterministe* M_1 qui reconnaît le même langage.

Autre formulation:

THÉORÈME: [Déterminisation]. Pour tout AFN M_0 défini sur S, il existe un AFD M_1 équivalent à M_0. Si M_0 a n états, alors M_1 a au plus 2^n états.

Preuve par construction

On pose $M_0 = (\Sigma, Q, I, F, \delta)$ et on considère M_1 défini par $M_1 = (\Sigma, 2^Q, \{I\}, F_1, \delta_1)$ avec :

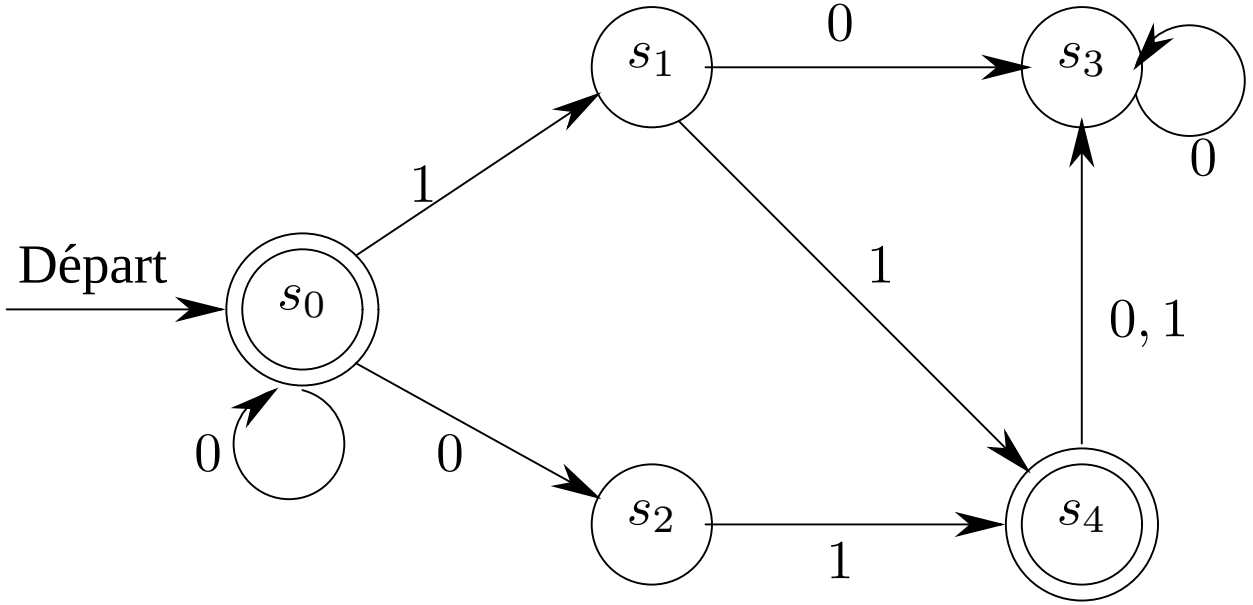
- $F_1 = \{G \subset Q, F \cap G \neq \varnothing\}$.
- $\delta_1(G, a) = \bigcup_{q \in G} \delta(q, a)$

Les états de M_1 sont donc associés de manière biunivoque à des sous-ensembles de Q.

- l'état initial est l'ensemble I ;
- chaque partie contenant un état final de M_0 donne lieu à un état final de M_1 ;
- la transition sortante d'un sous-ensemble E, étiquetée par a, atteint l'ensemble de tous les états de Q atteignables depuis un état de E par une transition étiquetée par a.
- M_1 est le déterminisé de M_0.

Exemple déterminisation d'un AFN (1)

Soit l'automate suivant à déterminiser:



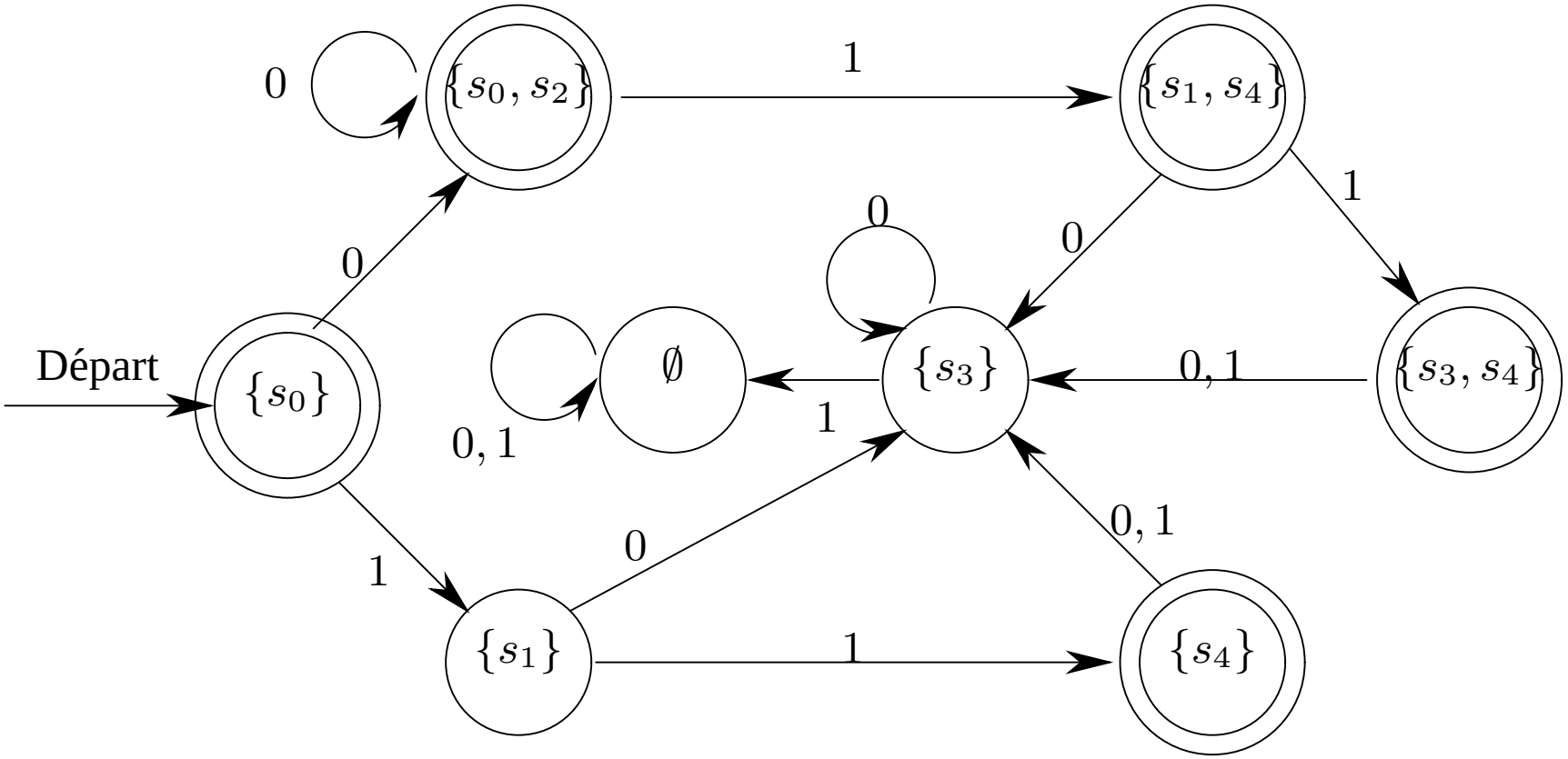
Exemple déterminisation d'un AFN (2)

	f	
S	I	
	0	1
s_0	$\{s_0, s_2\}$	$\{s_1\}$
s_1	$\{s_3\}$	$\{s_4\}$
s_2	$\varnothing$	$\{s_4\}$
s_3	$\{s_3\}$	$\varnothing$
s_4	$\{s_3\}$	$\{s_3\}$

	f	
S	I	
	0	1
s_0	$\{s_0, s_2\}$	$\{s_1\}$
s_1	$\{s_3\}$	$\{s_4\}$
s_2	$\varnothing$	$\{s_4\}$
s_3	$\{s_3\}$	$\varnothing$
s_4	$\{s_3\}$	$\{s_3\}$
s_0, s_2	$\{s_0, s_2\}$	$\{s_1, s_4\}$
s_1, s_4	$\{s_3\}$	$\{s_3, s_4\}$
s_3, s_4	$\{s_3\}$	$\{s_3\}$

Exemple déterminisation d'un AFN (3)

Résulat de la dégterminisation:



Automate fini minimal

DÉFINITION (Minimalité): Un automate $A = (Q, \Sigma, \delta, q_0, F)$ est minimal si pour tout $q, q' \in Q$, il existe $w \in L(q, A) \backslash L(q', A)$ (il n'y a pas deux états distincts qui acceptent le même langage).

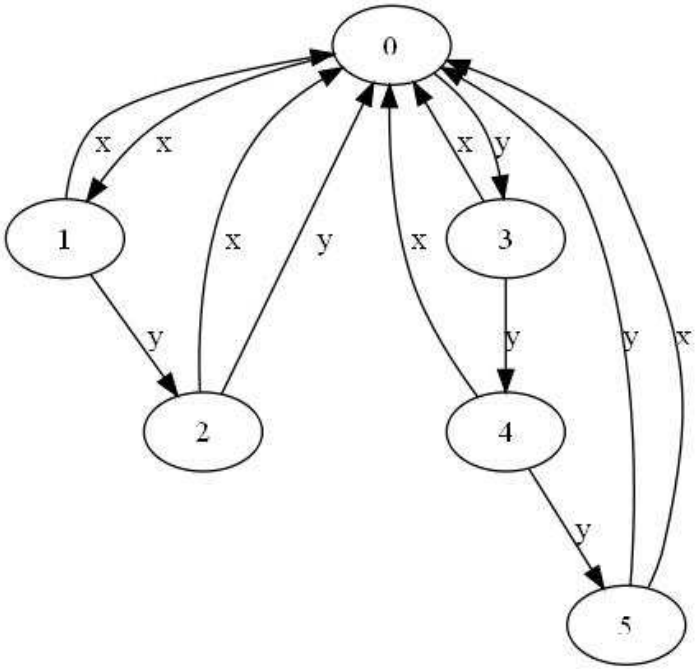
Autrement dit, la minimalité porte sur le nombre d'états de l'automate.

Minimisation d'un automate fini.

PRINCIPE : on définit des classes d'équivalence d'états par raffinements successifs. Chaque classe d'équivalence obtenue forme un seul même état du nouvel automate.

1. Faire deux classes : A contenant les états terminaux et B contenant les états non terminaux.
2. S'il existe un symbole a et deux états e_1 et e_2 d'une même classe tels que $\delta(e_1, a)$ et $\delta(e_2, a)$ n'appartiennent pas à la même classe, alors créer une nouvelle classe et séparer e_1 et e_2.
3. Recommencer 2 jusqu'à ce qu'il n'y ait plus de classes à séparer.
4. Chaque classe restante forme un état du nouvel automate

Exemple de minimisation d'un AF.



	x	y
0	1	3
1	0	2
2	0	0
3	0	4
4	0	5
5	0	0

$A = \{0\}, B = \{1, 2, 3, 4, 5\}$
$A = \{0\}, B = \{1, 3, 4\}, C = \{2, 5\}$
$A = \{0\}, B = \{1, 4\}, C = \{2, 5\}, D = \{3\}$

	x	y
A	B	D
B	A	C
C	A	A
D	A	B

Grammaires Régulières

Expressions et grammaires régulières

Définition: expression régulière

Les **expressions régulières** dans un ensemble I sont définies récursivement comme suit:

- Le symbole $\varnothing$ est une expression régulière,
- le symbole ϵ est une expression régulière,
- le symbole x est une expression régulière pour tout $x \in I$,
- Si A et B sont des expressions régulières, alors $(A.B)$, $(A \cup B)$ et A^* sont des expressions régulières.

Les expressions régulières dénotent des ensembles appelés **ensembles réguliers**.

Théorème de Kleene

THÉORÈME DE KLEENE: Un ensemble est régulier si et seulement si cet ensemble est reconnu par un automate fini.

DÉMONSTRATION : Un ensemble régulier est défini en fonction des expressions régulières qui sont définies récursivement. On peut prouver que tout ensemble régulier est reconnu par un automate fini si on peut démontrer que

1. $\varnothing$, $\{\epsilon\}$ et $\{a\}$ sont reconnus par un automate fini,
2. $A.B$ et $A \cup B$ sont reconnus par un automate fini si A et B le sont,
3. A^* est reconnu par un automate fini si A l'est.

On peut aisément construire tous ces automates ...

Démonstration du théorème de Kleene (suite 1)

Réciproquement : A partir d'un AFD A, on peut construire une ER qui dénote le langage reconnu par A.
Les systèmes d'équations linéaires à droite permettent de ramener le calcul d'une ER à la résolution d'un système d'équations.

- A chaque état q on associe une expression régulière Y_q dénotant le langage $L_q(A)$ associé à cet état. Y_q décrit le langage des mots tels que $\delta^*(q, w) \in F$ où δ^* est le prolongement de δ aux mots.
- On obtient un système d'équations dont les inconnues sont des ER dénotant des langages.
- Si q est l'état initial, Y_q décrit le langage reconnu par A.

Démonstration du théorème de Kleene (suite 2)

Pour résoudre le système obtenu on procède par substitutions et en résolvant l'équation : $Y = AY + B$

- A^*B est bien solution :

$$A^*B = AA^*B + B = A^+B + B = A^*B$$

- A^*B est une solution minimale : si Y est une solution, alors $A^*B \subseteq Y$
  Récurrence sur la hauteur d'étoile [a]:
  - si $i = 0$, $A^0B = B \subseteq Y$ car $Y = AY + B$
  - hyp. réc. pour $i = n$: $A^nB \subseteq Y$
  - pour $n + 1$: $A^{n+1}B = AA^nB \subseteq AY \subseteq AY + B = Y$, cqfd.

---

[a] définie par induction structurelle ($h(a) = h(\epsilon) = 0; h(e + f) = max(h(e), h(f)) = h(ef); h(e^*) = 1 + h(e)$)

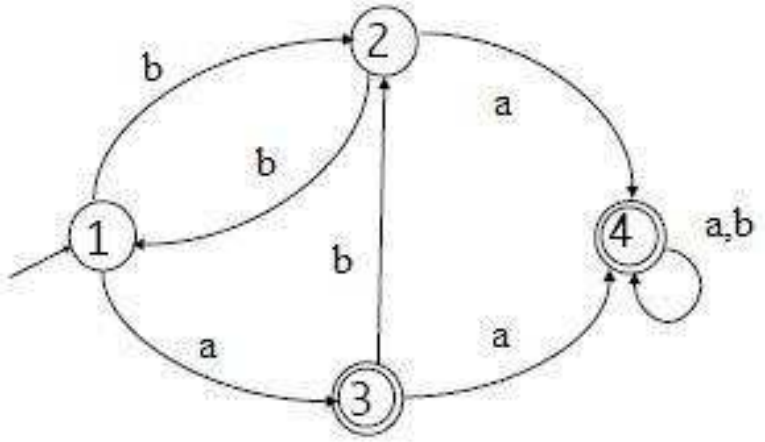
Démonstration du théorème de Kleene(suite 3)

- si $\epsilon \notin A$ alors A^*B est l'unique solution.
  On suppose la non-unicité de la solution A^*B.
  Soit X une autre solution et soit un mot w de longueur minimale tel que $w \in X \backslash A^*B$.
  $w \in X = AX + B$ et $w \notin B$ donc $w = uv$ avec $u \in A$ et $v \in X$. Or $v \notin A^*B$ donc $v \notin X\ A^*B$.
  Contradiction : la longueur de w n'était donc pas minimale !
- si $\epsilon \in A$ alors pour tout $C \subseteq \Sigma^*, Y = A^*B + A^*C$ est aussi solution :

$$A^*B + A^*C = A(A^*B + A^*C) + B = A^*B + A^+C = A^*B + A^*C$$

- si l'automate de départ est déterministe, on a toujours : $\epsilon \notin A$

Exemple



$$\begin{cases} Y_1 &= bY_2 + aY_3 \\ Y_2 &= bY_1 + aY_4 \\ Y_3 &= \epsilon + aY_4 + bY_2 \\ Y_4 &= \epsilon + (a+b)Y_4 \end{cases}$$

$$Y_4 = (a+b)^*$$

$$\begin{cases} Y_1 &= bY_2 + aY_3 \\ Y_2 &= bY_1 + a(a+b)^* \\ Y_3 &= \epsilon + a(a+b)^* + bY_2 \end{cases}$$

Exemple (suite)

$$\begin{cases} Y_1 &= bbY_1 + ba(a+b)^* + aY_3 \\ Y_3 &= \epsilon + a(a+b)^* + bbY_1 + ba(a+b)^* \end{cases}$$

D'où

$$\begin{cases} Y_1 &= bbY_1 + ba(a+b)^* + aY_3 \\ Y_3 &= \epsilon + bbY_1 + (b+\epsilon)a(a+b)^* \end{cases}$$

On en déduit:

$$Y_1 = bbY_1 + ba(a+b)^* + a\epsilon + abbY_1 + a(b+\epsilon)a(a+b)^*$$

Finalement:

$$Y_1 = (abb+bb)^*(ba(a+b)^* + a(\epsilon + (b+\epsilon)a(a+b)^*))$$

Rappel: grammaire régulière

Une grammaire régulière (ou de type 3)
$G = (V, T, S, P)$ est constituée d'un vocabulaire V,
d'un sous-ensemble T de V composé de terminaux,
d'un axiome S de V et d'un ensemble de
production P où chaque production a la forme:

- $S \rightarrow \epsilon$,
- $A \rightarrow a$,
- $A \rightarrow aB$,

où a est un symbole terminal et A et B sont des symboles non terminaux.

Grammaire régulière, ensemble régulier et automate fini

THÉORÈME: Un ensemble est généré par une grammaire régulière si et seulement s'il constitue un ensemble régulier.

DÉMONSTRATION: en exercice

De ce théorème "*grammaire régulière* $\Leftrightarrow$ *ensemble régulier*", du théorème de Kleene "*ensemble régulier* $\Leftrightarrow$ *automate fini*", on obtient par transitivité "*grammaire régulière* $\Leftrightarrow$ *automate fini*". Donc, si on a une grammaire régulière, alors il existe un automate fini que reconnaît le même langage, et vice-versa.

Transformation grammaire rég. $\rightarrow$ automate fini

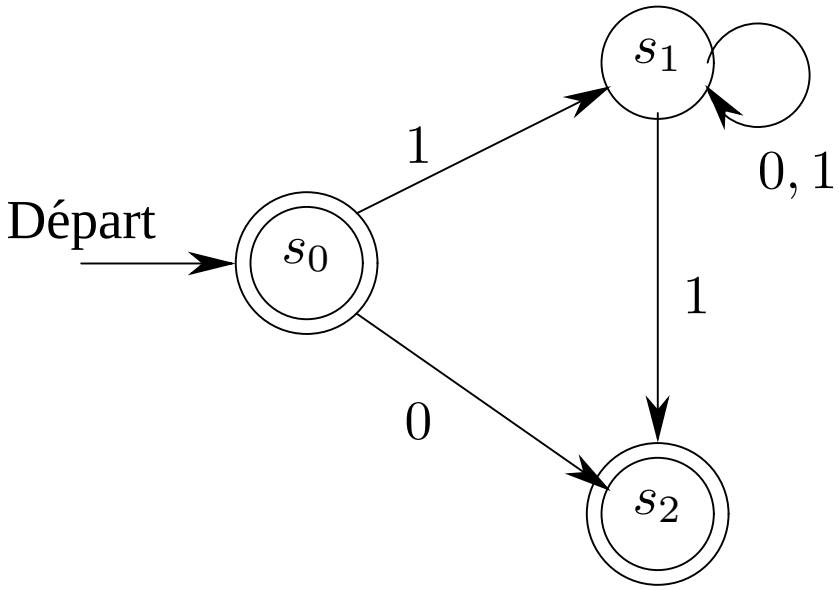
Soit $G = (V, T, S, P)$ une grammaire régulière générant l'ensemble $L(G)$.
On cherche l'automate fini $M = (S, I, f, s_0, F)$ qui reconnaît $L(G)$.

- S contient un état s_A pour chaque symbole non terminal A de G.
- S contient un état additionnel s_F qui est un état final.
- L'état de départ s_0 correspond au symbole de départ S.
- L'état s_0 est un état final si $S \rightarrow \epsilon$ est une production.
- Il y a une transition de s_A vers s_F à l'entrée de a si $A \rightarrow a$ est une production.
- Il y a une transition de s_A vers s_B à l'entrée de a si $A \rightarrow aB$ est une production.

Transformation grammaire rég. $\rightarrow$ automate fini

Soit $G = (V, T, S, P)$ où $V = \{0, 1, A, S\}$, $T = \{0, 1\}$ et les productions

- $S \rightarrow 1A$,
- $S \rightarrow 0$,
- $S \rightarrow \epsilon$,
- $A \rightarrow 0A$,
- $A \rightarrow 1A$,
- $A \rightarrow 1$.



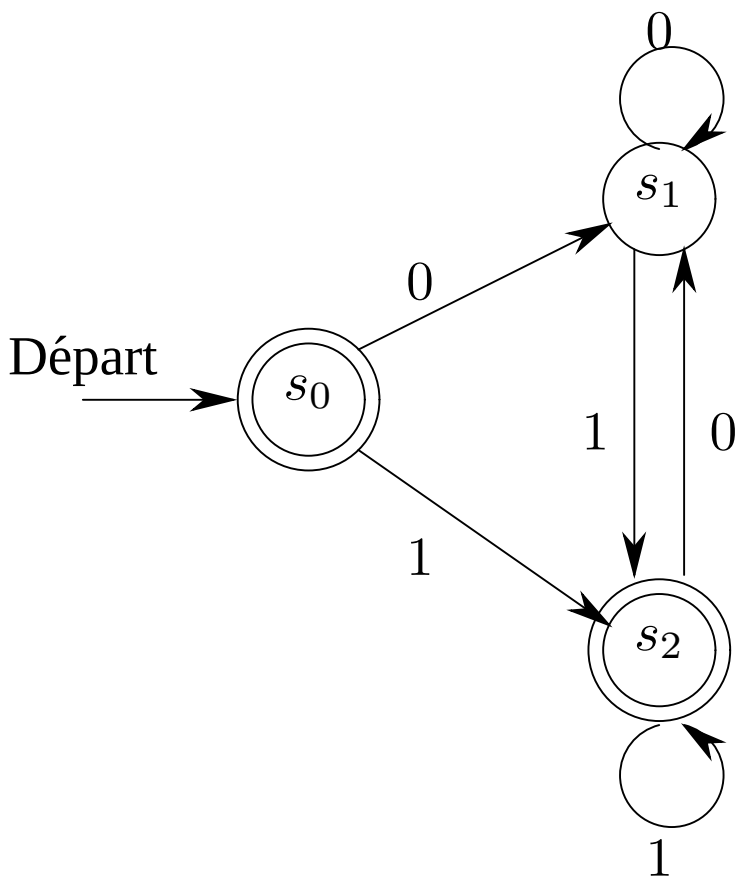
s_0 correspond à S, s_1 correspond à A et s_2 correspond à l'état final.

Transformation automate fini $\rightarrow$ grammaire rég.

Soit $M = (S, I, f, s_0, F)$ un AFD qui reconnaît $L(M)$. On cherche une grammaire régulière $G = (V, T, S, P)$ qui génère $L(M)$.

- V est formé en attribuant un symbole à chaque état de S et à chaque symbole de I.
- L'ensemble T des symboles terminaux correspond à I.
- Le symbole de départ S correspond à l'état de départ s_0.
- Si l'état s passe à un état final à l'entrée de a, alors $A_s \rightarrow a$ est une production.
- Si l'état s passe à l'état t à l'entrée de a, alors $A_s \rightarrow aA_t$ est une production.
- Si l'état de départ s_0 est un état final, alors $S \rightarrow \epsilon$ est une production.

Transformation automate fini $\rightarrow$ grammaire rég.



$G = (V, T, S, P)$ où $V = \{A, B, S, 0, 1\}$, $T = \{0, 1\}$, S, A et B correspondent aux états s_0, s_1 et s_2, et $P =$

- $S \rightarrow 0A$,
- $S \rightarrow 1B$,
- $S \rightarrow 1$,
- $S \rightarrow \epsilon$,
- $A \rightarrow 0A$,
- $A \rightarrow 1B$,
- $A \rightarrow 1$,
- $B \rightarrow 0A$,
- $B \rightarrow 1B$,
- $B \rightarrow 1$.

Limites des automates finis

Rappel: Un ensemble est reconnu par un automate fini si et seulement s'il est régulier.

On montre qu'il existe des ensembles qui ne sont pas réguliers en prouvant qu'ils ne sont pas reconnus par un automate fini.

Exemple: L'ensemble $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas régulier.

Voir la démonstration dans les slides suivants.

Lemme de pompage

Soit L un langage régulier reconnu par un automate à p états. Soit z un mot de L de longueur $\geq p$. Alors z se factorise en $z = uvw$, où $|uv| \leq p$, $v \neq \epsilon$ et, pour tout $i \geq 0$, on a $uv^iw \in L$.

DÉMONSTRATION

Si $z = a_1a_2...a_m$ est accepté par un automate à p états et que $m \geq p$, le chemin correspondant à z dans l'automate passe deux fois par un même état (principe des tiroirs). Soit q le premier état qui est revisité. On a alors $z = uvw$, où $\delta(q_0, u) = q$, et $\delta(q, v) = q$. On a $v \neq \epsilon$ et le chemin correspondant à uv visite l'état q deux fois et lui seul. Donc $|uv| \leq p$. Enfin, $\delta(q_0, uv^i) = q$ donc $\delta(q_0, uv^iw) = \delta(q_0, z)$ et finalement $uv^iw \in L$.

Une application du lemme de pompage

Il est souvent utilisé pour démontrer qu'un langage donné n'est pas régulier. La preuve se fait en général par l'absurde, en supposant que le langage est régulier et en exhibant un mot du langage de longueur suffisante tel que la propriété $uv^i w \in L$ ne soit pas vérifiée.

Exemple: Soit $L = \{a^n b^n\}$ sur l'alphabet $S = \{a, b\}$. Supposons par l'absurde que L est régulier. Soit p l'entier apparaissant le lemme. On choisit le mot $w = a^p b^p$. Il existe donc une décomposition $w = xyz$ vérifiant les conditions du lemme. $|xy| < p$ donc il existe deux entiers i et j tels que $i + j < p$ et $x = a^i, y = a^j$. De plus $j > 0$ car y n'est pas le mot vide. Il en résulte que $z = a^{p-i-j} b^p$. On devrait ainsi avoir $xz = xy^0 z = a^{p-j} b^p \ \in L$, ce qui est absurde car $p - j < p$. Donc L n'est pas régulier.

Le lemme est aussi vérifié pour de nombreux langages non-réguliers : il n'exprime donc qu'une condition nécessaire (mais pas suffisante).

Décidabilité

THÉORÈME : Les problèmes suivants sont décidables :

1. L'automate M reconnaît un langage non-vide ;
2. L'automate M reconnaît un langage infini ;
3. Le langage reconnu par M est inclus dans celui reconnu par N ;
4. Les automates M et N reconnaissent le même langage.

DÉMONSTRATION: en exercice