Langage rationnel
Automate Fini

Informatique Théorique 2
Licence 3 informatique

Sébastien Verel
verel@univ-littoral.fr
http://www-lisic.univ-littoral.fr/~verel

Université du Littoral Côte d'Opale
Laboratoire LISIC
Equipe OSMOSE

Les présentations

En gros mes recherches

PR. Laboratoire Informatique Signal et Image de la Côte d'Opale (LISIC)

Conception et étude d'algorithmes d'optimisation
(inspirés de la biologie)

- **Algorithmes** : méthode univoque ("recette de cuisine")
- **Optimisation** : trouver les meilleures solutions possibles à un problème (transport, emploi du temps, mobilité, énergie, agriculture,...)
- **Bio-inspiré** : principes actifs de systèmes biologiques : théorie de l'évolution, fouragement des fourmis, déplacement d'oiseaux, etc. (Systèmes Complexes, Intelligence Artificielle)
- **Conception** : créer et tester de nouveaux algorithmes
- **Etude** : comprendre et prédire pourquoi cela marche, ou mieux, pourquoi ça rate.

Informatique Théorique 2

- Fondement de l'informatique, bases avant compilation

- Volume : 27h
  - 22/09 et 24/09, 7,5h : Automate fini et autre
  - 13/10 et 15/10, 7,5h : Machine de Turing, calculabilité
  - 26/10, 27/10 et 29/10, 9,5h : Logique de Hoare
- Evaluation :
  - 20 % Interrogation 1, le 13/10/21 à 10h30, (45min),
  - 30 % Interrogation 2, le 26/10/21 à 8h30, (45min),
  - 50 % Examen final le 26/11/2021 à 9h00, (2h).

Web page de l'enseignement (supports et informations) :
`http://www-lisic.univ-littoral.fr/~verel`

Les systèmes formels

Utilisations pratiques

- Spécification des langages de programmation,
- Compilation, analyseur syntaxique,
- Recherche de motifs (texte, BD, web, etc.),
- les TALN
- Systèmes d'exploitation,
- Compression de texte,
- Cryptographie,
- Automatique,
- etc.

Aspects théoriques de l'Informatique

Questions fondamentales – et pratiques !

- **Spécification de programmes** :
  Sémantique d'un programme
- **Vérification de programmes** :
  Correction du programme
- **Complexité** :
  Temps, espace, énergie pour exécuter le calcul
- **Calculabilité** :
  Que peut-on calculer à l'aide d'un algorithme ?
  Quels problèmes peut-on résoudre avec un ordinateur ?

Aspects théoriques de l'Informatique

Questions fondamentales – et pratiques !

- **Spécification de programmes** :
  Sémantique d'un programme
- **Vérification de programmes** :
  Correction du programme
- **Complexité** :
  Temps, espace, énergie pour exécuter le calcul
- **Calculabilité** :
  Que peut-on calculer à l'aide d'un algorithme ?
  Quels problèmes peut-on résoudre avec un ordinateur ?

- Mot : réponse possible à un problème
- Langage : ensemble des réponses positives au problème
- Modèle de calcul : programme qui "calcule" le langage
  automate, machine de Turing, automate cellulaire, lambda-calcul, fonction récursive, etc.

Plan

Objectifs de la séance 01

Principalement des rappels :

- Savoir définir mot et langage
- Savoir écrire une expression régulière simple
- Connaitre les opérations algébriques sur les langages
- Savoir définir langage rationnel
- Savoir définir le langage reconnu par un automate fini déterministe
- Savoir construire un automate fini déterministe reconnaissant un langage
- Savoir écrire l'expression régulière du langage reconnu par un automate fini déterministe
- Connaitre la définition d'un automate fini non-déternimiste
- Savoir déterminiser un automate
- Savoir construire un automate à état fini reconnaissant un language rationnel simple
- Connaître le théorème de Kleene

Objectifs de la séance 01 (suite)

- Connaître le théorème de Kleene
- Savoir définir un automate fini déterministe
- Savoir définir un automate fini à partir d'une expression rationnelle
- Savoir construire un automate à partir des opérations algébriques sur les langages

Qu'est-ce qu'un mot ?

- Donner des exemples d'alphabet et de mots
- Ecrire la définition de mot
- Rappeler les définitions de : préfixe, suffixe, facteur, sous-mot, longueur, concaténation, etc.

Définition de mot

Alphabet

Un **alphabet** Σ est un ensemble *dénombrable*.

Les éléments de Σ sont appelés les **lettres** ou **symboles** de l'alphabet.

Remarque : très souvent un alphabet sera même un ensemble fini.

Définition de mot

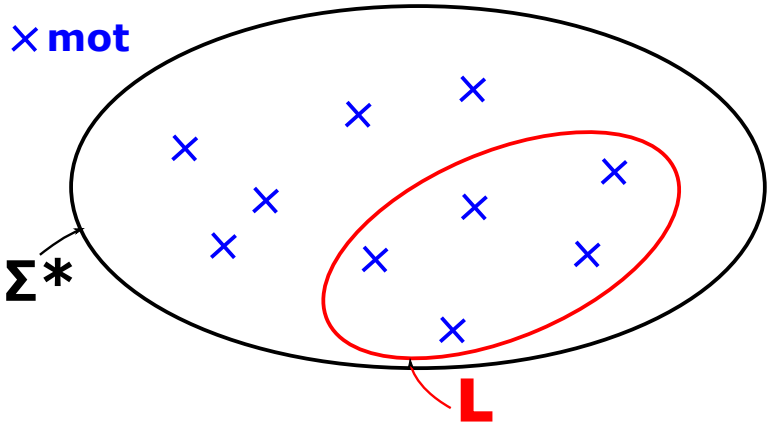
Mot

On appelle **mot** sur Σ toute suite *finie* de Σ :

$$u = (a_1, a_2, \ldots, a_n)$$

où $n \geq 0$, et pour tout $i \in [1..n]$, $a_i \in \Sigma$.
Lorsque $n = 0$, le mot est appelé le **mot vide**, noté ϵ.

Définir un langage



Σ Ensemble (fini) de lettres, alphabet

Σ^* Ensemble (dénombrable) des mots

Spécification de langage

- De manière informelle :

  mouais, pas très informatique, ambiguïté, à oublier.

- Par une **propriété caractéristique** P :

$$L_P = \{u \in \Sigma^* \ : \ P(u) \text{ vraie}\}$$

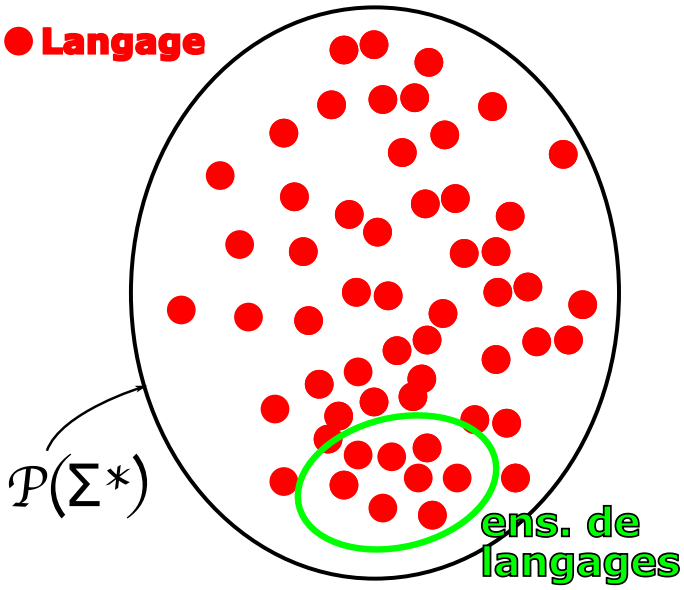
  Pas toujours un moyen pratique de déterminer si $u \in L$

- Par un algorithme A (**langage décidable**) :

$$L_A = \{u \in \Sigma^* \ : \ A(u) \text{ accept}\}$$

  Temps fini, moyen pratique pour savoir si $u \in L$

Ensemble de langages



Σ Ensemble (fini) de lettres, alphabet

Σ^* Ensemble (dénombrable) des mots

$\mathcal{P}(\Sigma^*)$ Ensemble de tous les langages

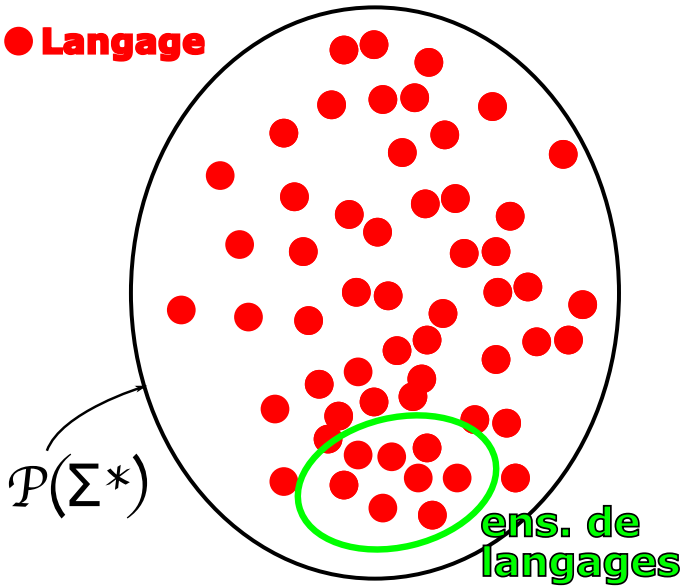
Ensemble de propriétés $\{P\}$ définissent un ensemble de langage :

$$LP = \{L \in \mathcal{P}(\Sigma^*) \ : \ \exists P \text{ telle que } L = L_P\}$$

Ensemble d'algorithmes $\{A\}$ définit un ensemble de langage :

$$LM = \{L \in \mathcal{P}(\Sigma^*) \ : \ \exists A \text{ telle que } L = L_A\}$$

Ensemble de langages



Σ Ensemble (fini) de lettres, alphabet

Σ^* Ensemble (dénombrable) des mots

$\mathcal{P}(\Sigma^*)$ Ensemble de tous les langages

Ensemble de propriétés $\{P\}$ définissent un ensemble de langage :

$$LP = \{L \in \mathcal{P}(\Sigma^*) \ : \ \exists P \text{ telle que } L = L_P\}$$

Ensemble d'algorithmes $\{A\}$ définit un ensemble de langage :

$$LM = \{L \in \mathcal{P}(\Sigma^*) \ : \ \exists A \text{ telle que } L = L_A\}$$

Tout l'art consiste à définir les bonnes propriétés et les bons modèles de machines...

Cardinalité de l'ensemble des langages

Cardinalité de $\mathcal{P}(\Sigma^*)$

L'ensemble des langages $\mathcal{P}(\Sigma^*)$ sur un alphabet Σ est non dénombrable.

Cardinalité de l'ensemble des langages

Cardinalité de $\mathcal{P}(\Sigma^*)$

L'ensemble des langages $\mathcal{P}(\Sigma^*)$ sur un alphabet Σ est non dénombrable.

Preuve :
Utilisation du procédé diagonal de Cantor avec la fonction caractéristique des langages.

Cardinalité de l'ensemble des langages

Cardinalité de $\mathcal{P}(\Sigma^*)$

L'ensemble des langages $\mathcal{P}(\Sigma^*)$ sur un alphabet Σ est non dénombrable.

Preuve :
Utilisation du procédé diagonal de Cantor avec la fonction caractéristique des langages.

Les conséquences sont **graves** !...

Expressions régulières (Rappel)

Rappel

- Les expressions régulières sont les expressions que l'on peut construire à partir de $+$, . et $*$ (cf. grep)

Conséquence

Une expression régulière définit un langage.

Par exemple

- $ab(a+b+c)$ est l'ensemble $\{aba, abb, abc\}$
- $(ab)^3$ est l'ensemble $\{ababab\}$
- $(ab)^*$: tous les mots qui répétent un nombre fini de fois ab (voir nul) $\{\epsilon, ab, abab, ababab, \ldots\}$
- $ab(a+b+c)^*$: tous les mots qui comment par ab

Opérations rationnelles sur les langages

Soient deux langages L et M respect. sur les alphabets Σ_L et Σ_M. L et M sont des ensembles, il est possible d'appliquer les opérations ensemblistes.

- **union** et **intersection** : (langages sur l'alphabet $\Sigma_L \cup \Sigma_M$)

$$L \cup M \text{ et } L \cap M$$

Opérations rationnelles sur les langages

Soient deux langages L et M respect. sur les alphabets Σ_L et Σ_M. L et M sont des ensembles, il est possible d'appliquer les opérations ensemblistes.

- **union** et **intersection** : (langages sur l'alphabet $\Sigma_L \cup \Sigma_M$)

$$L \cup M \text{ et } L \cap M$$

- **complémentaire** :

$$L^c = \Sigma_L^* - L$$

Opérations rationnelles sur les langages

Soient deux langages L et M respect. sur les alphabets Σ_L et Σ_M. L et M sont des ensembles, il est possible d'appliquer les opérations ensemblistes.

- **union** et **intersection** : (langages sur l'alphabet $\Sigma_L \cup \Sigma_M$)
$$L \cup M \text{ et } L \cap M$$
- **complémentaire** :
$$L^c = \Sigma_L^* - L$$
- **produit/concaténation** (langage sur l'alphabet $\Sigma_L \cup \Sigma_M$) :
$$LM = \{uv \mid u \in L \ , \ v \in M\}$$
- **puissance** :
  - $L^0 = \{\epsilon\}$
  - pour $i > 0$, $L^i = L.L^{i-1}$

Opérations rationnelles sur les langages

Soient deux langages L et M respect. sur les alphabets Σ_L et Σ_M. L et M sont des ensembles, il est possible d'appliquer les opérations ensemblistes.

- **union** et **intersection** : (langages sur l'alphabet $\Sigma_L \cup \Sigma_M$)
$$L \cup M \text{ et } L \cap M$$
- **complémentaire** :
$$L^c = \Sigma_L^* - L$$
- **produit/concaténation** (langage sur l'alphabet $\Sigma_L \cup \Sigma_M$) :
$$LM = \{uv \mid u \in L \ , \ v \in M\}$$
- **puissance** :
  - $L^0 = \{\epsilon\}$
  - pour $i > 0$, $L^i = L.L^{i-1}$
- **étoile** :
$$L^* = \cup_{i \geq 0} L^i$$

Exemples

Soient $L = \{pl, dl, pdpl, ldl\}$ et $M = \{\epsilon, ai, ia\}$

- $L \cup M =$

Exemples

Soient $L = \{pl, dl, pdpl, ldl\}$ et $M = \{\epsilon, ai, ia\}$

- $L \cup M = \{pl, dl, pdpl, ldl, \epsilon, ai, ia\}$
- $L \cap M =$

Exemples

Soient $L = \{pl, dl, pdpl, ldl\}$ et $M = \{\epsilon, ai, ia\}$

- $L \cup M = \{pl, dl, pdpl, ldl, \epsilon, ai, ia\}$
- $L \cap M = \emptyset$
- $L^c = \Sigma^* - L$

Exemples

Soient $L = \{pl, dl, pdpl, ldl\}$ et $M = \{\epsilon, ai, ia\}$

- $L \cup M = \{pl, dl, pdpl, ldl, \epsilon, ai, ia\}$
- $L \cap M = \emptyset$
- $L^c = \Sigma^* - L$: trop long....
- $LM =$

Exemples

Soient $L = \{pl, dl, pdpl, ldl\}$ et $M = \{\epsilon, ai, ia\}$

- $L \cup M = \{pl, dl, pdpl, ldl, \epsilon, ai, ia\}$
- $L \cap M = \emptyset$
- $L^c = \Sigma^* - L$: trop long....
- $LM = \{pl, dl, pdpl, ldl,$
  $plai, dlai, pdplai, ldlai,$
  $plia, dlia, pdplia, ldlia, \}$
- $M^2 =$

Exemples

Soient $L = \{pl, dl, pdpl, ldl\}$ et $M = \{\epsilon, ai, ia\}$

- $L \cup M = \{pl, dl, pdpl, ldl, \epsilon, ai, ia\}$
- $L \cap M = \emptyset$
- $L^c = \Sigma^* - L$: trop long....
- $LM = \{pl, dl, pdpl, ldl,$
  $plai, dlai, pdplai, ldlai,$
  $plia, dlia, pdplia, ldlia, \}$
- $M^2 = \{\epsilon, ai, ia, aiai, iaia, aiia, iaai\}$
- étoile :

$$M^* = \cup_{i \geq 0} M^i$$

...

Langage rationnel

Définition inductive de l'ensemble des langages rationnels (ou réguliers) :

Langages rationnels (ou réguliers)

L'ensemble des langages rationnels LR est défini par :

- *base :*
  - $\emptyset \in LR$
  - $\{\epsilon\} \in LR$
  - pour tout $a \in \Sigma$, $\{a\} \in LR$.
- *induction :* Si $L \in LR$ et $M \in LR$ alors :
  - $L \cup M \in LR$
  - $L.M \in LR$
  - $L^* \in LR$.

Remarque :
LR est clos par réunion, concaténation et $*$.

Expression Régulière (ER) et Langage Rationnel (LR)

Théorème (admis)

Un langage est rationnel (ou régulier)
si et seulement si
il est définit par une expression régulière.

d'où le nom de langage rationnel / régulier...

Introduction

Spécifier un langage L :

- donner une description des mots qui sont dans L (ER, etc.),

ou, de façon équivalente :

- donner un algorithme permettant de décider si un mot quelconque est, ou n'est pas dans L.

Les automates sont des **machines abstraites** programmables qui permettent de reconnaitre si un mot appartient, ou non, à un langage L.

Ils existent dans de nombreuses machines concrètes (machine à café, robots, ...)

Exemple

Définir un automate qui autorise les séquences possibles d'un distributeur de machine à café avec :
café, eau, lait, sucre, annulation, payer, etc.

Définition

Automate fini déterministe

Un **automate fini déterministe** (AFD) est un quintuplet (Q, Σ, T, q_0, A) où :

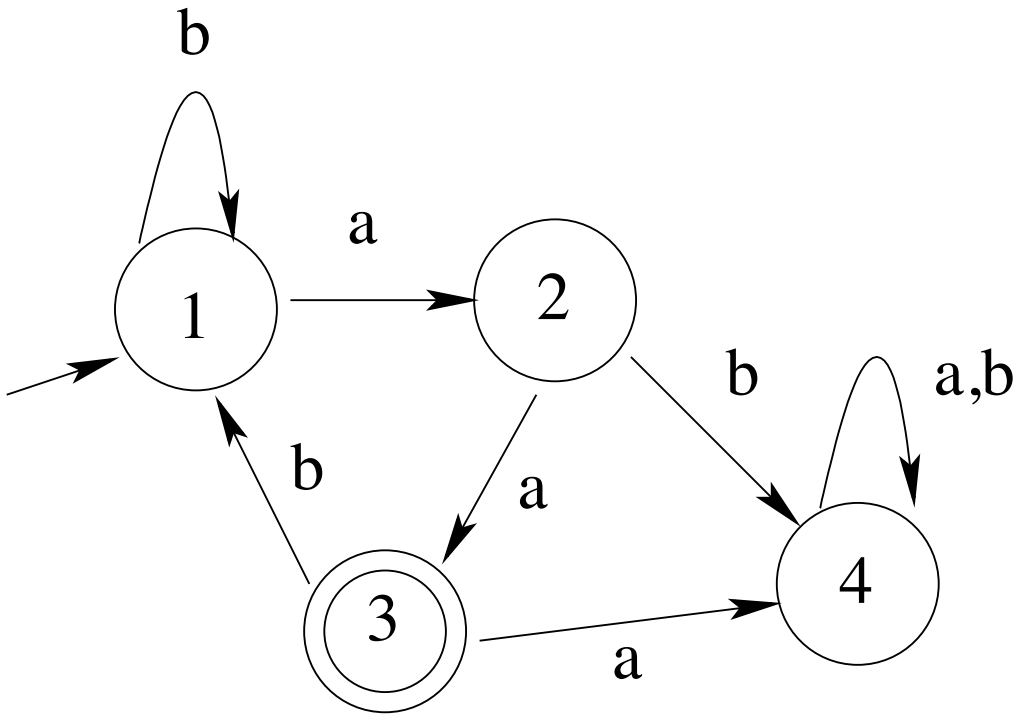
- Σ est l'*alphabet* de l'automate,
- Q un ensemble **fini** appelé *ensemble des états* de l'automate,
- T est une application de $Q \times \Sigma$ dans Q, appelée la *fonction de transition*
- q_0 est un élément de Q, appelé l'*état initial*
- A est un sous-ensemble de Q, appelé l'ensemble des *états acceptants*.

Exemples

$\Sigma = \{a, b\}$,
$Q = \{1, 2, 3, 4\}$,
$q_0 = 1$ est l'état initial
$A = \{3\}$ est l'ensemble des états acceptants.

T

		a	b
$\rightarrow$	1	2	1
	2	3	4
	$\boxed{3}$	4	1
	4	4	4



Calcul sur automate : Lecture

fonction de transition itérée

La **fonction de transition itérée** est l'application $T^* : Q \times \Sigma^* \to Q$ définie par :

- *base :* $T^*(q, \epsilon) = q$
- *induction :* Pout tout $w_0 \in \Sigma^*$ et $\sigma \in \Sigma$, $T^*(q, w_0\sigma) = T(T^*(q, w_0), \sigma)$

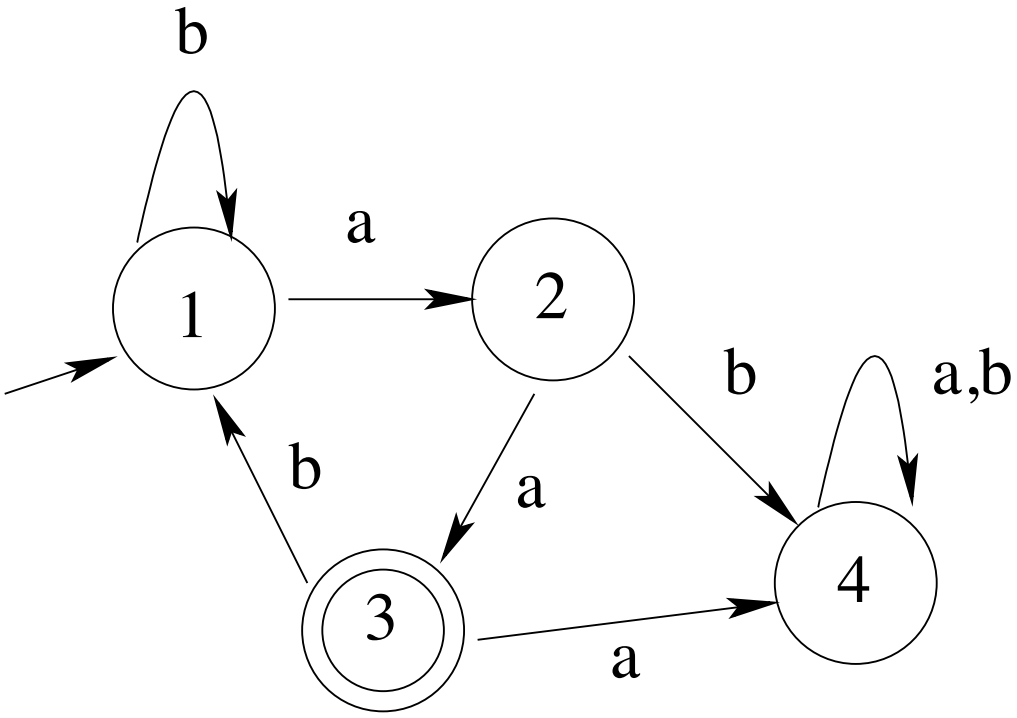
La fonction de transition itérée donne l'état final après la lecture du w par l'automate.

Exemples

$\Sigma = \{a, b\}$,
$Q = \{1, 2, 3, 4\}$,
$q_0 = 1$ est l'état initial
$A = \{3\}$ est l'ensemble des états acceptants.

T

		a	b
$\rightarrow$	1	2	1
	2	3	4
	3	4	1
	4	4	4



lecture de *baabbbaa*

Langage décidé

Mot accepté / refusé

Soit l'automate $M = (Q, \Sigma, T, q_0, A)$.

Un mot $w \in \Sigma^*$ est **accepté** par M ssi $T^*(q_0, w) \in A$.
Un mot $w \in \Sigma^*$ est **refusé** par M ssi $T^*(q_0, w) \notin A$.

Langage décidé

Soient M est un automate d'alphabet Σ et L un langage sur Σ

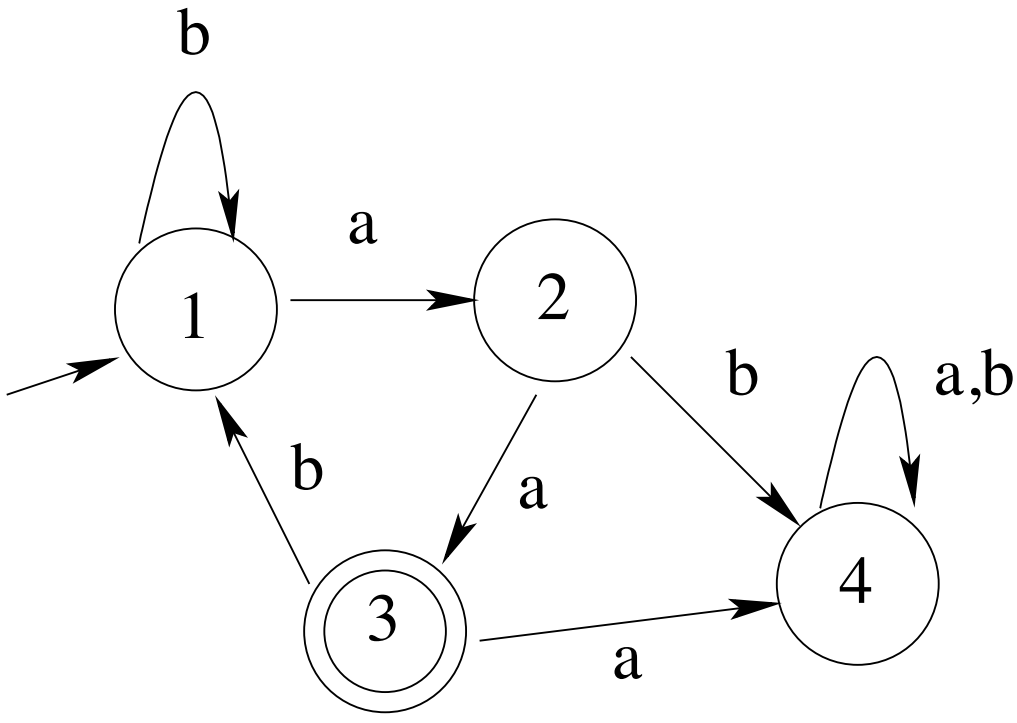
M **décide** L ssi L est l'ensemble des mots acceptés par M.

Exemples

$\Sigma = \{a, b\}$,
$Q = \{1, 2, 3, 4\}$,
$q_0 = 1$ est l'état initial
$A = \{3\}$ est l'ensemble des états acceptants.

T

		a	b
$\rightarrow$	1	2	1
	2	3	4
	$\boxed{3}$	4	1
	4	4	4



Remarque finale

L'**état** d'un automate est la **mémoire** de la machine,

l'état de l'automate sert à mémoriser ...
...l'état de la lecture du mot (ce qui vient d'être lu)

Le nombre d'état étant fini,
il n'est possible de mémoriser qu'un nombre fini de lettres lues.

Introduction

Pour pouvoir définir un automate qui reconnait un langage rationnel,

Il faudrait définir un automate qui puisse reconnaitre :

- la réunion de langages,
- la concaténation de langages,
- l'étoile d'un langage (fermeture de Kleene).

Introduction de non-déterministes

Non-déterministe

En informatique, **non-déterministe** est souvent associé à "plusieurs choix possibles" par opposition **déterministe** où l'opération ou l'action à effectuer est unique, *i.e.* complètement déterminé par l'état actuel du système (sans ambiguité).

Introduction de non-déterministes

Non-déterministe

En informatique, **non-déterministe** est souvent associé à "plusieurs choix possibles" par opposition **déterministe** où l'opération ou l'action à effectuer est unique, *i.e.* complètement déterminé par l'état actuel du système (sans ambiguité).

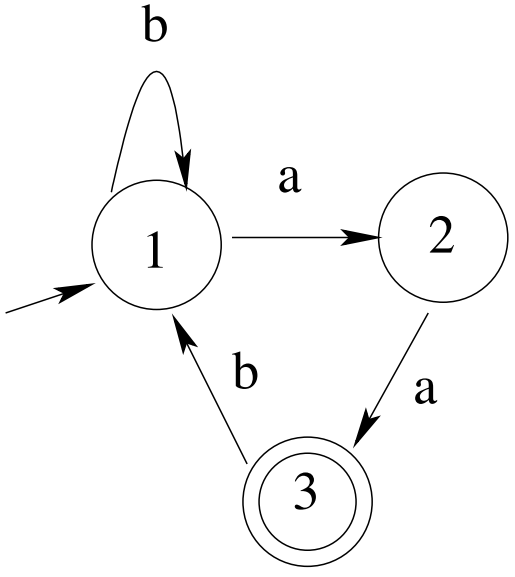
Non-déterministe dans les automates

Plusieurs sources de non-déterministe dans les automates :

- Absence de transition,
- Plusieurs transitions pour une même lettre,
- Plusieurs états initiaux,
- des transitions sur des mots vides : ϵ-transitions.

Absence de transition

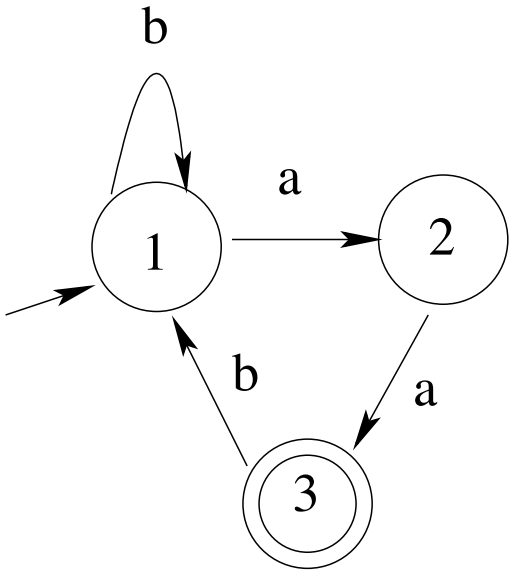
lecture de *bbba*



S'il n'y a plus de transition possible
et que le mot est encore en cours de lecture
Alors le mot est refusé

Absence de transition

lecture de *bbba*

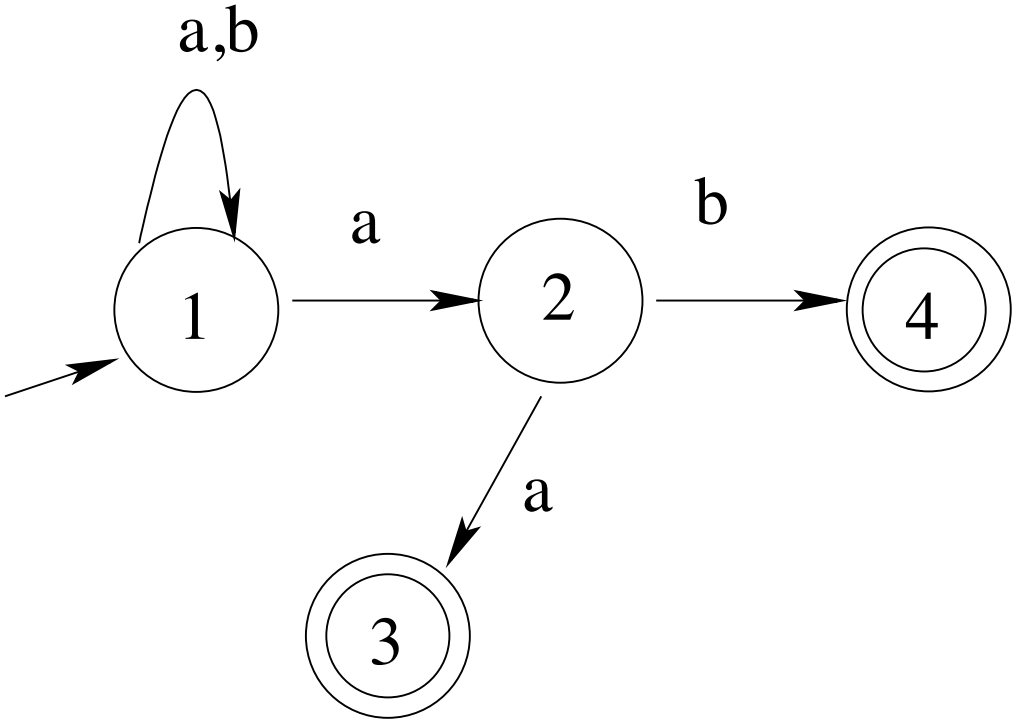


S'il n'y a plus de transition possible
et que le mot est encore en cours de lecture
Alors le mot est refusé

Remarque : remplace la technique de l'état "puit"

Plusieurs transitions

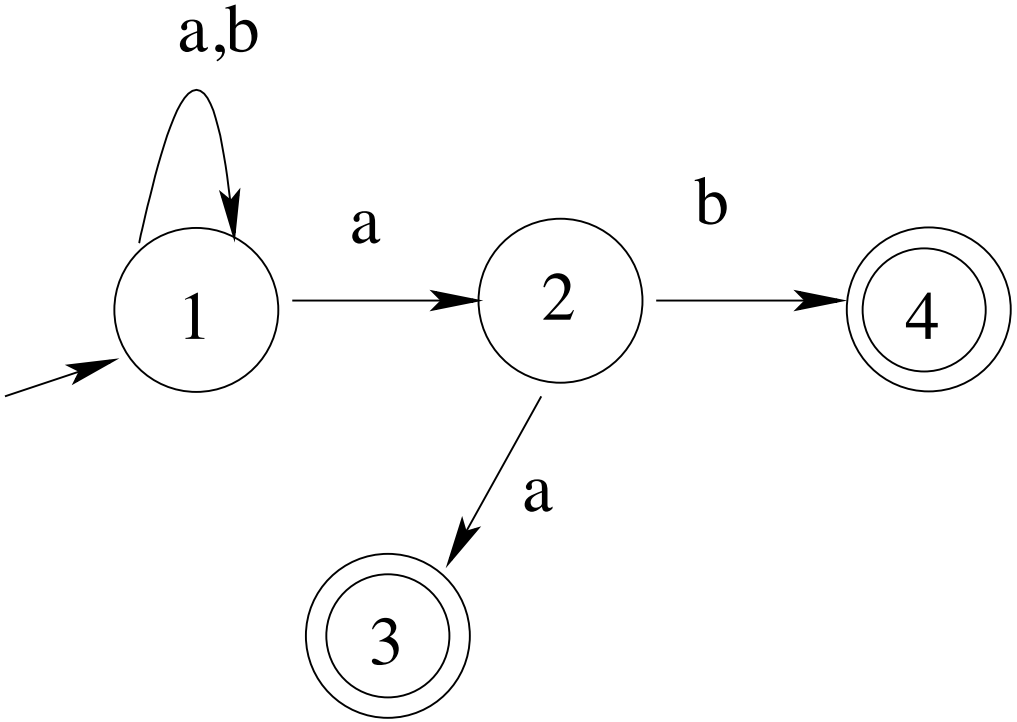
Lecture de *aabaabab* et de *aaaaaba*



Le mot est accepté lorsqu'il existe au moins une lecture menant à un état acceptant.

Plusieurs transitions

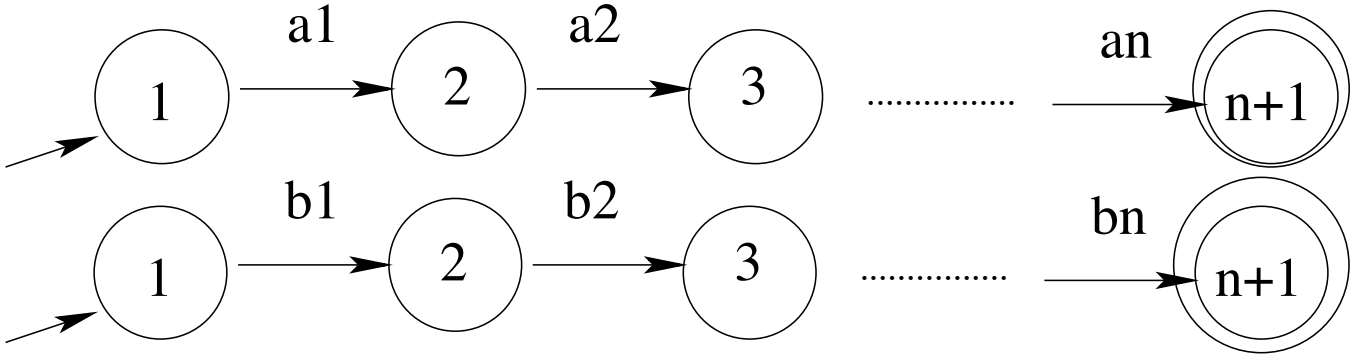
Lecture de *aabaabab* et de *aaaaaba*



Le mot est accepté lorsqu'il existe au moins une lecture menant à un état acceptant.

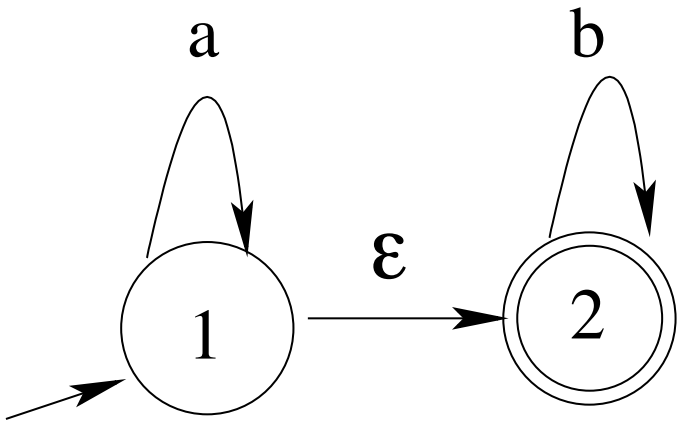
Conséquence : Il faut essayer tous les lectures possibles pour être sûr que le mot est refusé

Plusieurs états initiaux



Le mot est accepté lorsqu'il existe une lecture à partir de l'un des états initiaux menant à un état acceptant.

ϵ-transistions



Une ϵ-transistion est une transition par lecture du mot vide.

Pendant la lecture d'un mot, il est possible de choisir d'effectuer la transition ϵ sans lire aucune lettre.

Définition AFN

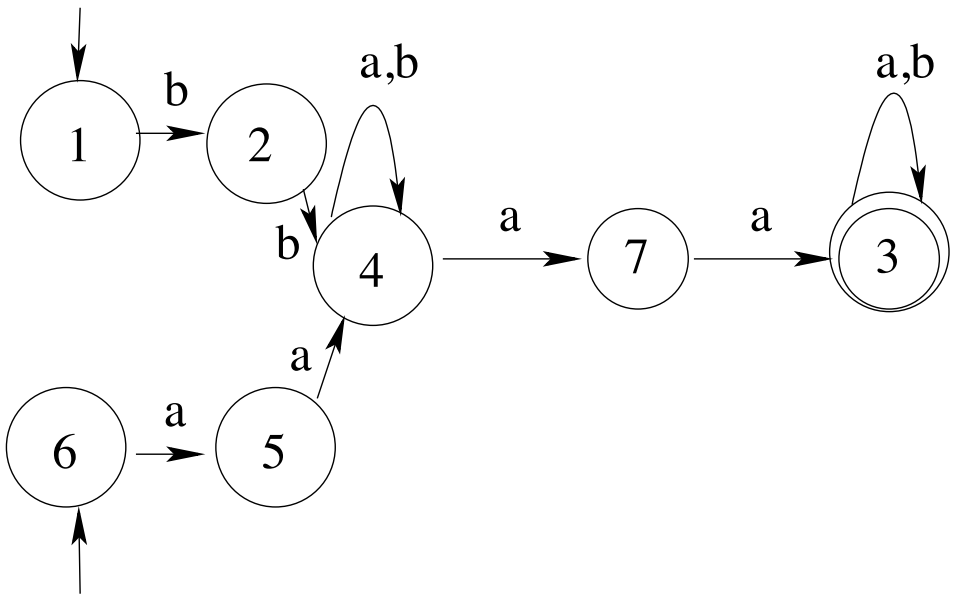
Automate Fini Non-déterministe (AFN)

Un **Automate Fini Non-déterministe** est un quintuplet (Q, Σ, T, I, A) où :

- Σ est l'*alphabet* de l'automate,
- Q un ensemble fini appelé *ensemble des états* de l'automate,
- T est une application de $Q \times \Sigma$ dans $\mathcal{P}(Q)$, appelée la *fonction de transition*
- I est un sous-ensemble de Q, appelé l'ensemble *des états initiaux*
- A est un sous-ensemble de Q, appelé l'*ensemble des états acceptants*.

Exemple

		a	b
→	1	-	2
	2	-	4
	3	3	3
	4	4,7	4
	5	4	-
→	6	5	-
	7	3	-



Lecture / reconnaissance

Lecture

Soient $M = (Q, \Sigma, T, I, A)$ un AFN et $u = x_1 x_2 \dots x_l$ un mot sur Σ.

Une lecture de u par M est une suite d'états $(q_0, q_1, \dots, q_l)$ vérifiant :

i $q_0 \in I$, et

ii $q_i \in T(q_{i-1}, x_i)$ pour $1 \leq i \leq l$.

Lecture / reconnaissance

Lecture

Soient $M = (Q, \Sigma, T, I, A)$ un AFN et $u = x_1 x_2 \dots x_l$ un mot sur Σ.

Une lecture de u par M est une suite d'états $(q_0, q_1, \dots, q_l)$ vérifiant :

i $q_0 \in I$, et

ii $q_i \in T(q_{i-1}, x_i)$ pour $1 \leq i \leq l$.

Acceptation

Le mot u est accepté par M s'il existe au moins une lecture de u par M qui se termine par un état acceptant.

Equivalence déterministe / non-déterministe

Définition équivalence

Soient M et M' deux automates. On dit que M et M' sont équivalents s'ils acceptent et refusent exactement les mêmes mots.

Equivalence déterministe / non-déterministe

Définition équivalence

Soient M et $M^{'}$ deux automates. On dit que M et $M^{'}$ sont équivalents s'ils acceptent et refusent exactement les mêmes mots.

Equivalence : Déterministe $\Rightarrow$ Non-déterministe

L'automate déterministe $M = (Q, \Sigma, T, q_0, A)$ est équivalent à l'automate non-déterministe $M = (Q, \Sigma, T^{'}, \{q_0\}, A)$ avec $T^{'}(q, x) = \{T(q, x)\}$.

Equivalence déterministe / non-déterministe

Equivalence : Non-déterministe ⇒ Déterministe (admis)

Soient $M = (Q, \Sigma, T, I, A)$ un AFN. Alors M est équivalent l'AFD M' définit par $M' = (\mathcal{P}(Q), \Sigma, T', I, A')$ avec :

- $T'(X, x) = \cup_{q \in X} T(q, x)$
- $A' = \{X \in \mathcal{P}(Q) \mid X \cap A \neq \emptyset\}$

Remarques :

- Un état dans l'automate déterministe est un ensemble.
- Un état pour M' est acceptant lorsqu'il contient un état acceptant pour M.

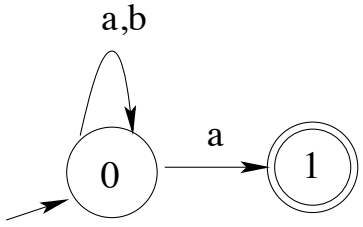
Algorithme de déterminisation

Les mots se terminant par a :

Algorithme de déterminisation

Les mots se terminant par a :

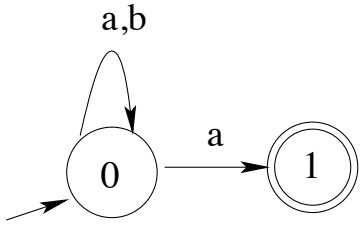
		a	b
$\rightarrow$	0	0,1	0
	$\boxed{1}$	-	-



Algorithme de déterminisation

Les mots se terminant par a :

		a	b
$\rightarrow$	0	0,1	0
	1	-	-



AFD équivalent :

		a	b
$\rightarrow$	0	0,1	0
	0,1	0,1	0

On part de l'état initial et pour chaque état suivant, on réunit l'ensemble des états atteignables depuis cet état.

Algorithme de déterminisation

		a	b
$\rightarrow$	0	0	2
$\rightarrow$	1	2	1
	$\boxed{2}$	-	-

On regroupe les états initiaux dans un même ensemble :

Algorithme de déterminisation

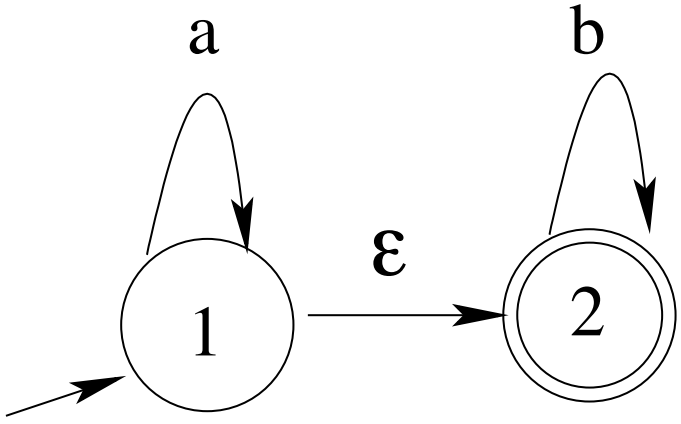
		a	b
$\rightarrow$	0	0	2
$\rightarrow$	1	2	1
	$\boxed{2}$	-	-

On regroupe les états initiaux dans un même ensemble :

		a	b
$\rightarrow$	0,1	0,2	1,2
	$\boxed{0,2}$	0	2
	$\boxed{1,2}$	2	1
	0	0	2
	$\boxed{2}$	$\square$	$\square$
	1	2	1
	$\square$	$\square$	$\square$

$\square$ est un état "puit"

Et les ϵ-transistions ?



Une ϵ-transistion est une transition sur un mot vide.

Pendant la lecture d'un mot, il est possible de choisir d'effectuer la transition ϵ sans lire aucune lettre.

Définition $\Sigma_@$

Définir les ϵ-transitions consiste à définir un alphabet $\Sigma_@$ où il existe une lettre supplémentaire correspondant à ϵ.

Définition de $\Sigma_@$

$$\Sigma_@ = \Sigma \cup \{@\}$$

Notons $\pi_@ : \Sigma_@^* \to \Sigma^*$ la fonction (projection) qui remplace :

- chaque lettre de Σ par elle-même
- @ par le mot vide.

Exemple

Si $\Sigma = \{a, b\}$ alors $\Sigma_@ = \{a, b, @\}$
et $\pi_@(@aa@b@@b) = aabb$

Remarque : @ représente le mot vide ϵ, $\pi_@(u)$ sous-mot de u

Définition AFN_ϵ

AFN_ϵ

Un **Automate Fini Non-déterministe avec ϵ-transitions** est un quintuplet (Q, Σ, T, I, A) où :

- Σ est l'*alphabet* de l'automate,
- Q un ensemble fini appelé *ensemble des états* de l'automate,
- T est une application de $Q \times \Sigma_@$ dans $\mathcal{P}(Q)$, appelée la *fonction de transition*
- I est un sous-ensemble de Q, appelé l'*ensemble des états initiaux*
- A est un sous-ensemble de Q, appelé l'*ensemble des états acceptants*.

Acceptation

Acceptation

Un mot u sur Σ est accepté par l'AFN$_\epsilon$ (Q, Σ, T, I, A) s'il existe au moins un mot $u_@$ sur $\Sigma_@$ qui est accepté par l'AFN $(Q, \Sigma_@, T, I, A)$ et tel que $u = \pi_@(u_@)$.

Intuitivement, un mot est accepté s'il existe un parcours de l'automate avec ϵ-transitions "spontanées"

Equivalence AFN_ϵ / AFN

Equivalence (admis)

Soient $M = (Q, \Sigma, T, I, A)$ un AFN_ϵ. Alors M est équivalent l'AFN M' définit par $M' = (Q, \Sigma, T', I', A')$ avec :

- $T'(q, x) = \cup_{q' \in cl(q)} T(q', x)$
- $I' = \cup_{q \in I} cl(q)$
- $A' = \{q \mid cl(q) \cap A \neq \emptyset\}$

$cl(q)$ est la clôture (union des itérés) de q par ϵ-transitions, c'est-à-dire l'ensemble des états atteignables par ϵ-transitions itérées (cf. suite).

Equivalence AFN_ϵ / AFN

Cloture de q

$cl(q)$ est la cloture de q par ϵ-transitions, c'est-à-dire l'ensemble des états atteignables par ϵ-transitions itérées.

Cloture : Définition ascendante

$$cl(q) = \cup_{i \in \mathbb{N}} X_i$$

avec :

- $X_0 = \{q\}$
- $X_{i+1} = X_i \cup \{q' : q' = T(q_i, @) \text{ avec } i \in X_i\}$

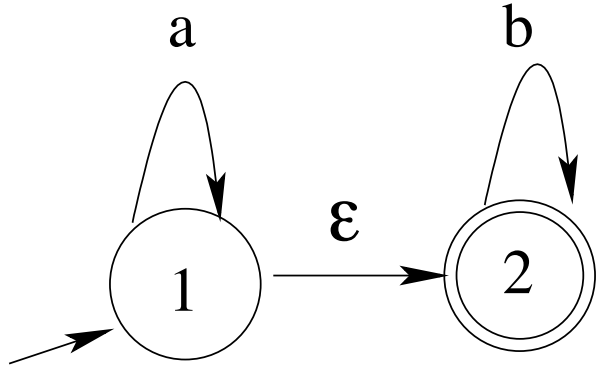
Cloture : Définition descendante

$$cl(q) = \cap\{X \ : \ q \in X \text{ et } X \text{ stable par } \epsilon\text{-transistion}\}$$

On dit que $X \subset \mathcal{P}(Q)$ est stable par ϵ-transition si $T(X, @) \subset X$.

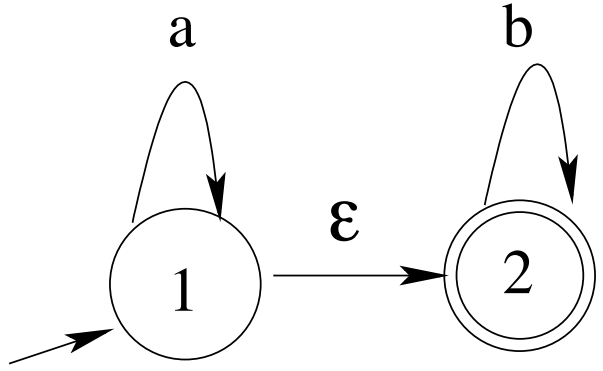
Algorithme de déterminisation

		a	b	ϵ
$\rightarrow$	1	1	-	2
	2	-	2	-



Algorithme de déterminisation

		a	b	ϵ
$\rightarrow$	1	1	-	2
	2	-	2	-



AFN équivalent :

$cl(1) = \{1, 2\}$

		a	b
$\rightarrow$	**1**	1	2
$\rightarrow$	**2**	-	2

Théorème de Kleene

Théorème de Kleene (admis...)

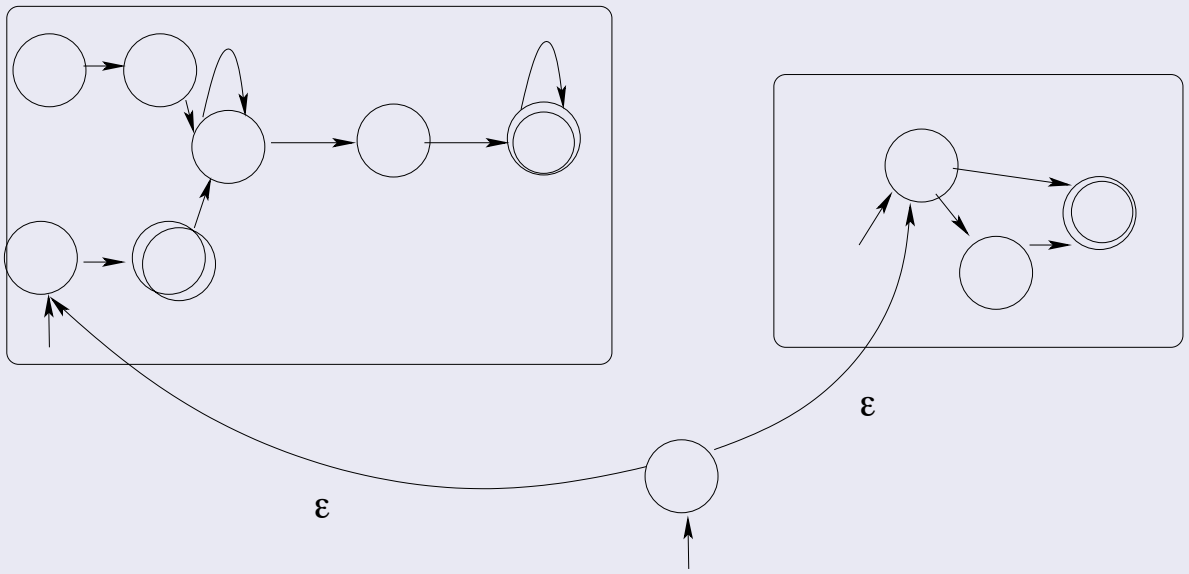
Un langage sur un alphabet Σ est rationnel si et seulement si il est reconnu par un automate fini.

Idée de la démonstration :
On peut construire de manière inductive l'ensemble des langages rationnels et les automates reconnaissant ces langages.

Union de langages rationnels

Soient deux automates finis déterministes M_1 et M_2 reconnaissant respectivement les langages L_1 et L_2

$L_1 \cup L_2$ est reconnu par :

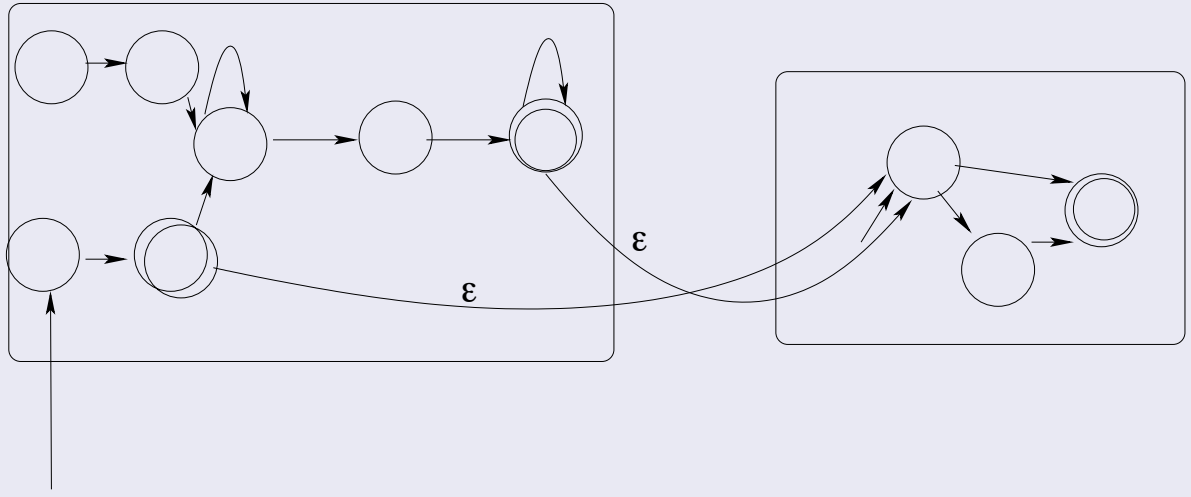


On ajoute des ϵ-transitions entre un nouvel état initial et les états initiaux de M_1 et de M_2

Produit de concaténation de langages rationnels

Soient deux automates finis déterministes M_1 et M_2 reconnaissant respectivement les langages L_1 et L_2

$L_1.L_2$ est reconnu par :

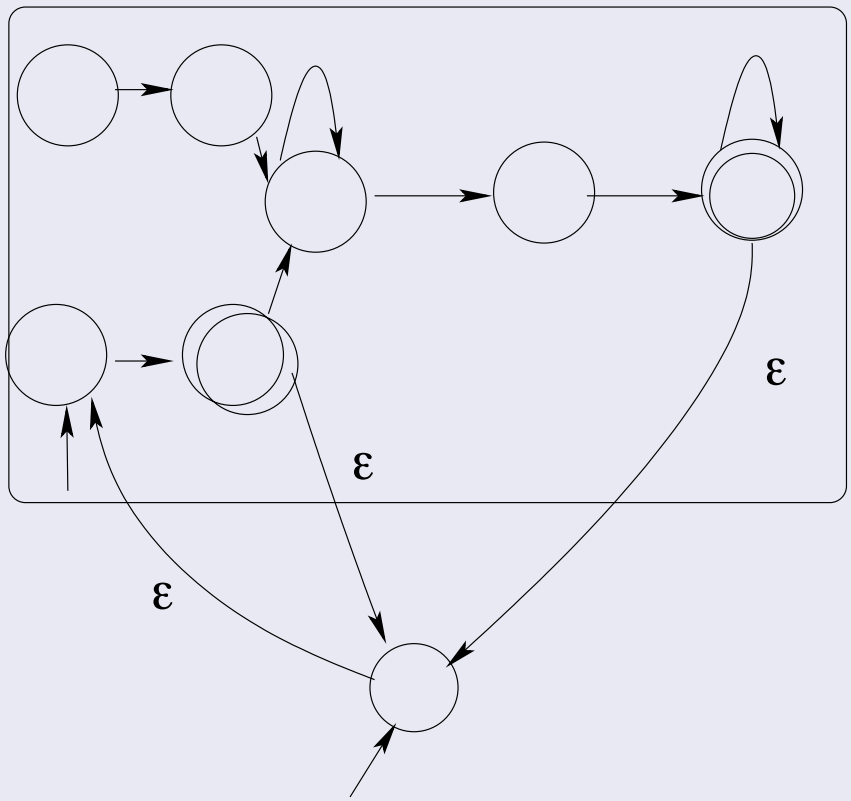


On ajoute des ϵ-transitions entre les états acceptants de M_1 et l'état intial de M_2

Etoile (cloture de Kleene) de langages rationnels

Soit un automate fini déterministe M reconnaissant le langage L

L^* est reconnu par :



On ajoute des ϵ-transitions entre les états finaux et le nouvel état initial

Conclusion (1)

- A chaque **langage rationnel** est associé
  un **automate fini**, et réciproquement.
- Les automates sont des machines abstraites capables de réaliser des calculs sur des mots :
  entrée : mot (donnée du problème)
  sortie : oui/non (une décision)
- Lien très fort entre langage et machine :
  - Langage : définit un ensemble de mots
  - Machine : calcul un ensemble de mots

Conclusion (2)

- Il est possible de définir d'autres machines abstraites qui permettent de définir d'autres classes de langages.

  L'expressivité du langage et la capacité de calcul de la machine sont alors différentes.
- Les questions que l'on se pose sont alors les mêmes :
  - mode de lecture,
  - description algébrique langage (souvent à l'aide d'une définition inductive),
  - équivalence avec d'autres classes de langages,
  - complexité de calcul d'une machine reconnaissant le langage.

Par exemple, on peut remplacer automate par machine de Turing...