Langages et automates

DUT informatique

IUT d'Arles

2017 — 2018

Exemples

Automates finis

Langages

Opérations sur les langages

Langage d'un automate fini

Langages réguliers

Création d'automates

Sommaire

Pair/Impair

- La société Pair/Impair est spécialisé dans les mots binaires

Pair/Impair

- La société Pair/Impair est spécialisé dans les mots binaires
- Lorsqu'on lui soumet un mot, par exemple,

  $$0001001011101010111010101111001$$

  elle détermine si le nombre de bits égaux à 1 est pair.

Pair/Impair

- La société Pair/Impair est spécialisé dans les mots binaires
- Lorsqu'on lui soumet un mot, par exemple,

  $$0001001011101010111010101111001$$

  elle détermine si le nombre de bits égaux à 1 est pair.
- elle employait jusqu'à présent une personne qui comptait le nombre de bits égaux à 1 et qui regardait si ce nombre était pair

Pair/Impair

- Cette personne est malheureusement partie à la retraite

Pair/Impair

- Cette personne est malheureusement partie à la retraite
- Rigueur budgétaire aidant, elle est remplacée par quelqu'un de si peu qualifié qu'il ne sait même pas compter !

Pair/Impair

- Cette personne est malheureusement partie à la retraite
- Rigueur budgétaire aidant, elle est remplacée par quelqu'un de si peu qualifié qu'il ne sait même pas compter !

Que faire ?

Pair/Impair

Le nouveau ne sait pas compter mais . . .

Pair/Impair

Le nouveau ne sait pas compter mais . . .

- il sait distinguer un 0 d'un 1 ;

Pair/Impair

Le nouveau ne sait pas compter mais . . .

- il sait distinguer un 0 d'un 1 ;
- et il n'est pas manchot !

Pair/Impair

Le nouveau ne sait pas compter mais . . .

- il sait distinguer un 0 d'un 1 ;
- et il n'est pas manchot !

« Voici ce que tu va faire :

Pair/Impair

Le nouveau ne sait pas compter mais . . .

- il sait distinguer un 0 d'un 1 ;
- et il n'est pas manchot !

« Voici ce que tu va faire :

- Au départ, tu lèves ton pouce et tu passes en revue tous les bits

Pair/Impair

Le nouveau ne sait pas compter mais . . .

- il sait distinguer un 0 d'un 1 ;
- et il n'est pas manchot !

« Voici ce que tu va faire :

- Au départ, tu lèves ton pouce et tu passes en revue tous les bits
- À chaque fois que tu rencontres un 0, tu ne fais rien

Pair/Impair

Le nouveau ne sait pas compter mais . . .

- il sait distinguer un 0 d'un 1 ;
- et il n'est pas manchot !

« Voici ce que tu va faire :

- Au départ, tu lèves ton pouce et tu passes en revue tous les bits
- À chaque fois que tu rencontres un 0, tu ne fais rien
- À chaque fois que tu rencontres un 1, tu change la position de ton pouce »

Pair/Impair

Quand tu as fini,

Pair/Impair

Quand tu as fini,

- Si ton pouce est tourné vers le haut alors, le nombre de bits égaux à 1 est pair

Pair/Impair

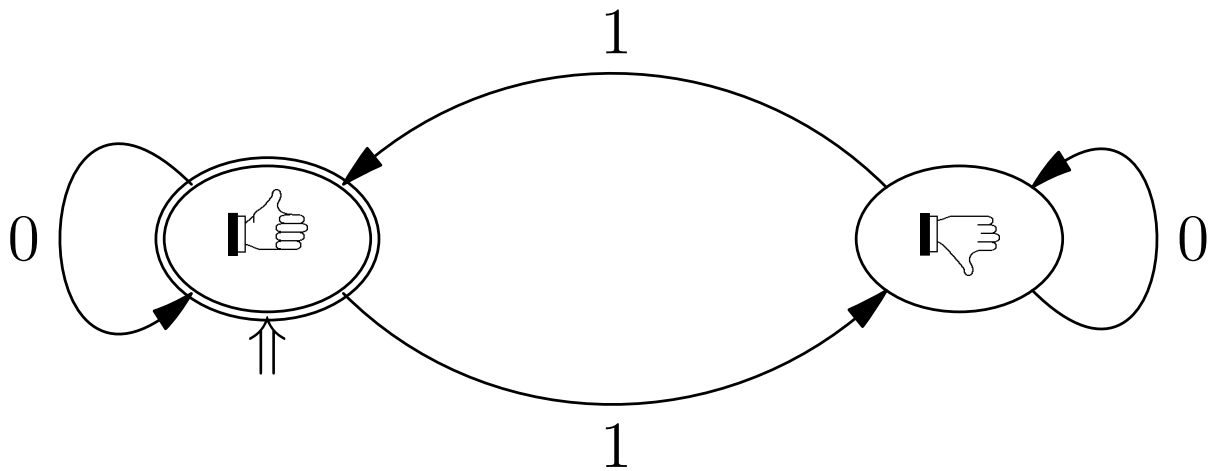
Quand tu as fini,

- Si ton pouce est tourné vers le haut alors, le nombre de bits égaux à 1 est pair
- Sinon il est impair

Pair/Impair

Quand tu as fini,

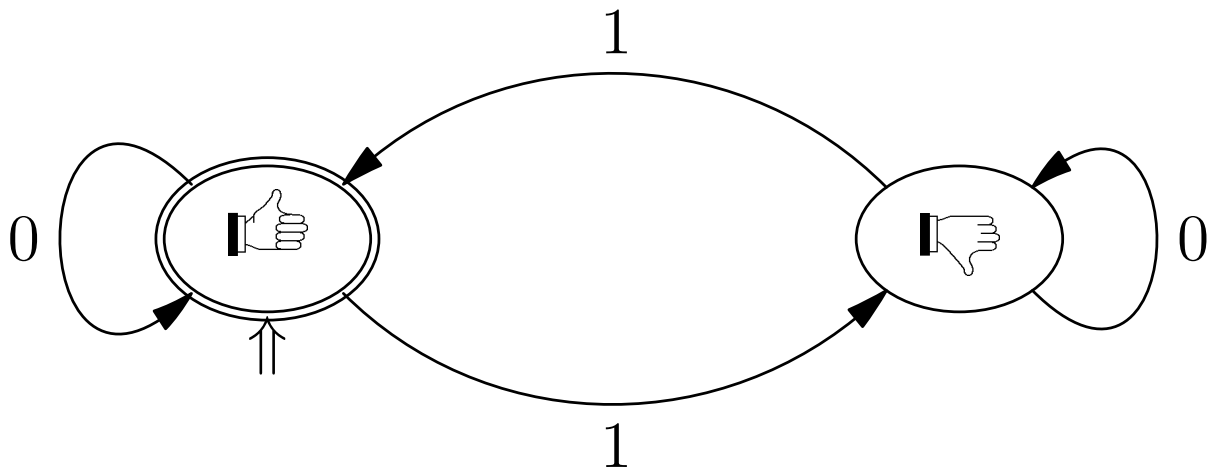
- Si ton pouce est tourné vers le haut alors, le nombre de bits égaux à 1 est pair
- Sinon il est impair



Pair/Impair

Quand tu as fini,

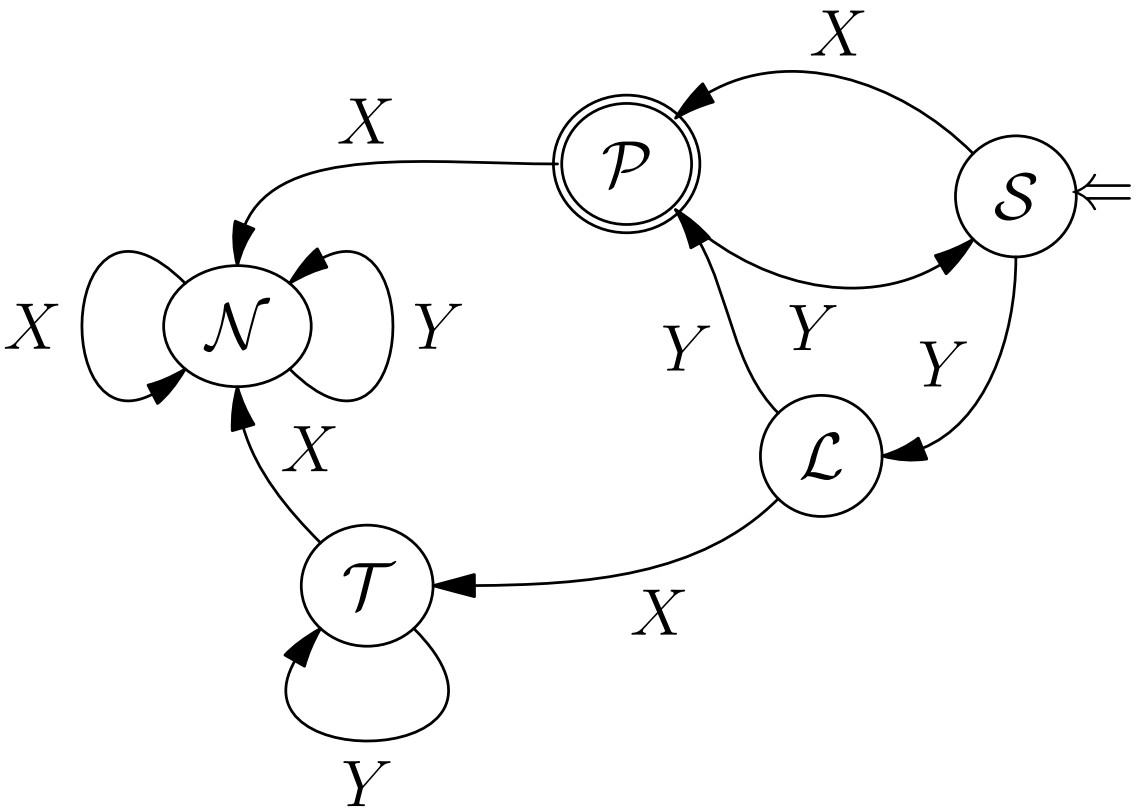
- Si ton pouce est tourné vers le haut alors, le nombre de bits égaux à 1 est pair
- Sinon il est impair



Le nouveau fit les gestes représentés et trouva que le nombre de bits égaux à 1 dans 0110 est pair . . .

Jeu

Ce matin, j'ai reçu un courrier publicitaire avec une carte

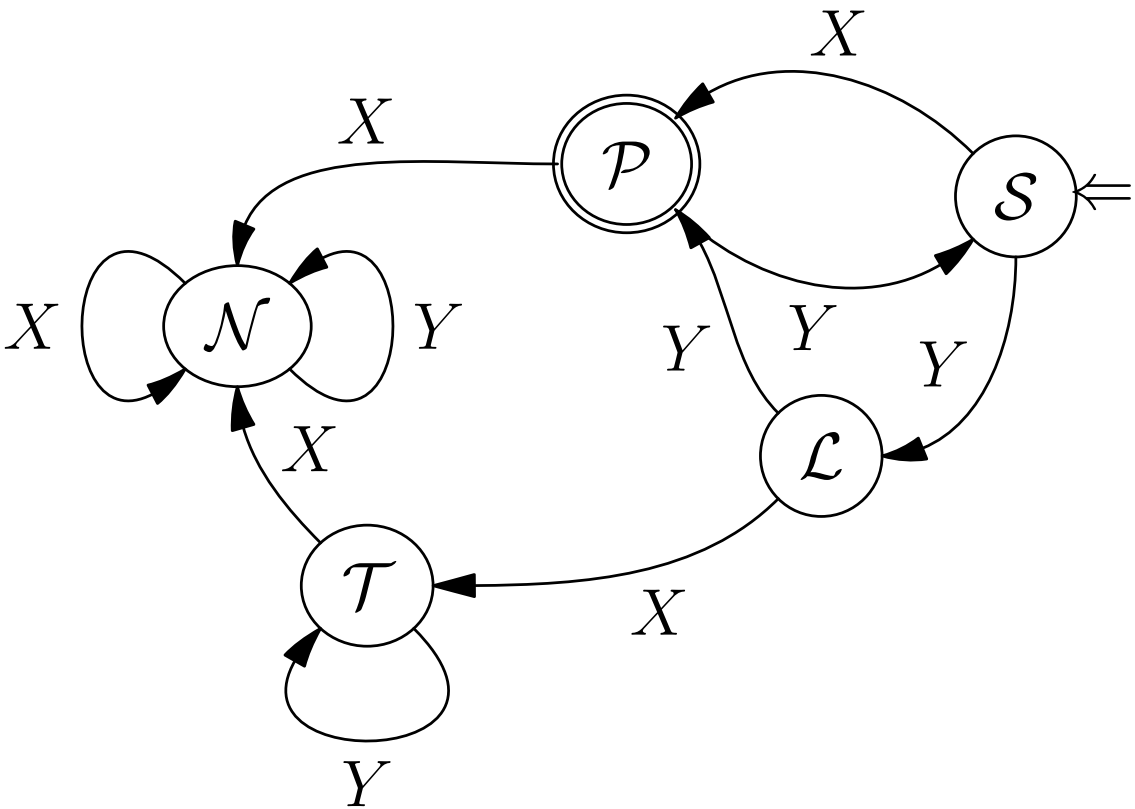


$\mathcal{L}$yon
$\mathcal{N}$antes
$\mathcal{P}$aris
$\mathcal{S}$trasbourg
$\mathcal{T}$oulouse

.

Jeu

Ce matin, j'ai reçu un courrier publicitaire avec une carte

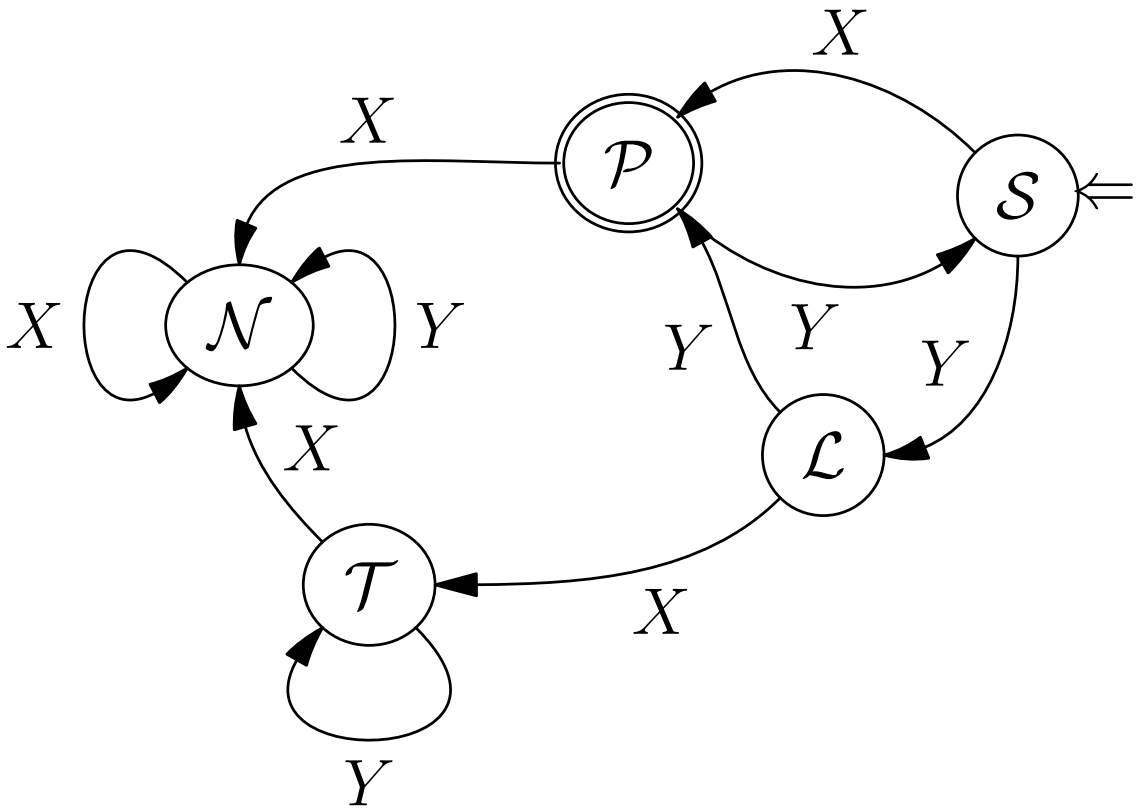


$\mathcal{L}$yon
$\mathcal{N}$antes
$\mathcal{P}$aris
$\mathcal{S}$trasbourg
$\mathcal{T}$oulouse

et un code confidentiel $\mathcal{YYXY}$.

Jeu

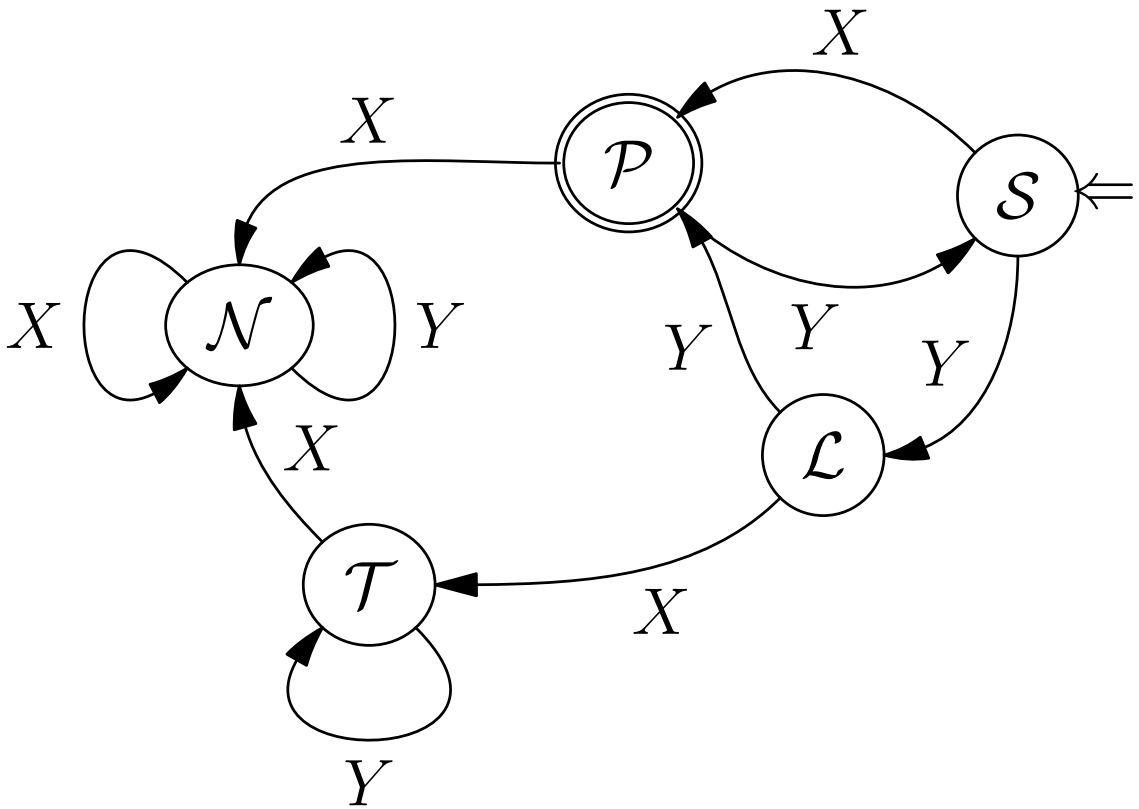
Séjour offert d'une semaine dans toute les villes du parcours correspondant au code confidentiel au départ de Strasbourg . . .



$\boxed{\mathcal{YYXY}}$

Jeu

Séjour offert d'une semaine dans toute les villes du parcours correspondant au code confidentiel au départ de Strasbourg . . .



$\boxed{\mathcal{YYXY}}$

. . . à condition d'être à Paris la dernière semaine

Automates finis

- Les deux diagrammes précédents représentent des *automates finis*

Automates finis

- Les deux diagrammes précédents représentent des *automates finis*
- On peut les voir comme une machine dans laquelle on introduit une *chaîne de caractères*

Automates finis

- Les deux diagrammes précédents représentent des *automates finis*
- On peut les voir comme une machine dans laquelle on introduit une *chaîne de caractères*
  - 0110 pour le premier
  - $\mathcal{Y}\mathcal{Y}\mathcal{X}\mathcal{Y}$ pour le second

Automates finis

- Les deux diagrammes précédents représentent des *automates finis*
- On peut les voir comme une machine dans laquelle on introduit une *chaîne de caractères*
  - 0110 pour le premier
  - $\mathcal{Y}\mathcal{Y}\mathcal{X}\mathcal{Y}$ pour le second
- Une fois le dernier caractère introduit, l'automate donne une réponse sous la forme de *oui* ou *non*

Automates finis

- Les deux diagrammes précédents représentent des *automates finis*
- On peut les voir comme une machine dans laquelle on introduit une *chaîne de caractères*
  - 0110 pour le premier
  - $\mathcal{YYXY}$ pour le second
- Une fois le dernier caractère introduit, l'automate donne une réponse sous la forme de *oui* ou *non*
  - oui, 0110 possède un nombre pair de 1
  - non, vous n'avez pas gagné

Automates finis

- On introduit le 1^{er} caractère

Automates finis

- On introduit le 1^{er} caractère
- l'introduction de chaque nouveau caractère induit un mouvement

Automates finis

- On introduit le 1^{er} caractère
- l'introduction de chaque nouveau caractère induit un mouvement
- quand le dernier caractère a été introduit, on regarde sur quelle position on s'est arrêté

Automates finis

- On introduit le 1^{er} caractère
- l'introduction de chaque nouveau caractère induit un mouvement
- quand le dernier caractère a été introduit, on regarde sur quelle position on s'est arrêté
- pour certaines positions déterminées à l'avance la réponse de l'automate est *oui*

Automates finis

- On introduit le 1^{er} caractère
- l'introduction de chaque nouveau caractère induit un mouvement
- quand le dernier caractère a été introduit, on regarde sur quelle position on s'est arrêté
- pour certaines positions déterminées à l'avance la réponse de l'automate est *oui*
- pour toutes les autres, la réponse est non

Si la réponse est oui :

Automates finis

Si la réponse est oui :

- on dit que la chaîne de caractère est *acceptée* ou *reconnue* par l'automate (sinon elle est *refusée*)

Automates finis

Si la réponse est oui :

- on dit que la chaîne de caractère est *acceptée* ou *reconnue* par l'automate (sinon elle est *refusée*)
- L'ensemble des chaînes acceptées est le *langage accepté* par l'automate ou plus simplement le *langage de l'automate*

Automates finis

Si la réponse est oui :

- on dit que la chaîne de caractère est *acceptée* ou *reconnue* par l'automate (sinon elle est *refusée*)
- L'ensemble des chaînes acceptées est le *langage accepté* par l'automate ou plus simplement le *langage de l'automate*

Il reste à formaliser tout ça d'un point de vue mathématique

Sommaire

Définition

Définir un automate fini $\mathcal{A}$ c'est se donner :

Définition

Définir un automate fini $\mathcal{A}$ c'est se donner :

- Un ensemble fini non vide Σ : l'*alphabet* de l'automate

Définition

Définir un automate fini $\mathcal{A}$ c'est se donner :

- Un ensemble fini non vide Σ : l'*alphabet* de l'automate
- Un ensemble fini non vide $\mathcal{E}$: les *états* de l'automate

Définition

Définir un automate fini $\mathcal{A}$ c'est se donner :

- Un ensemble fini non vide Σ : l'*alphabet* de l'automate
- Un ensemble fini non vide $\mathcal{E}$: les *états* de l'automate
- Un état particulier $I \in \mathcal{E}$: l'état initial de l'automate

Définition

Définir un automate fini $\mathcal{A}$ c'est se donner :

- Un ensemble fini non vide Σ : l'*alphabet* de l'automate
- Un ensemble fini non vide $\mathcal{E}$: les *états* de l'automate
- Un état particulier $I \in \mathcal{E}$: l'état initial de l'automate
- Une partie $A \subset \mathcal{E}$ constituée des *états acceptants*

Définition

Définir un automate fini $\mathcal{A}$ c'est se donner :

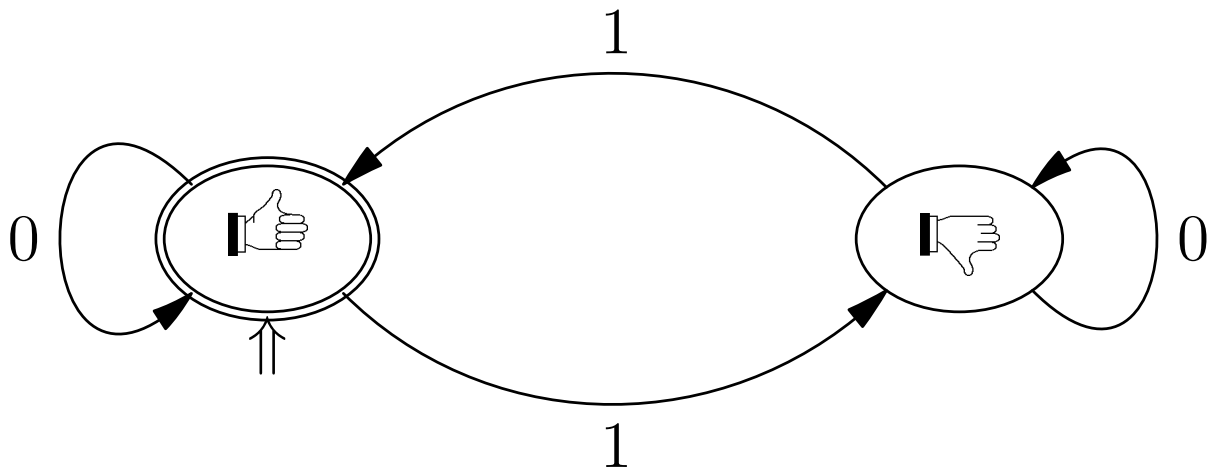
- Un ensemble fini non vide Σ : l'*alphabet* de l'automate
- Un ensemble fini non vide $\mathcal{E}$: les *états* de l'automate
- Un état particulier $I \in \mathcal{E}$: l'état initial de l'automate
- Une partie $A \subset \mathcal{E}$ constituée des *états acceptants*
  Les autres sont les états *refusant*

Définition

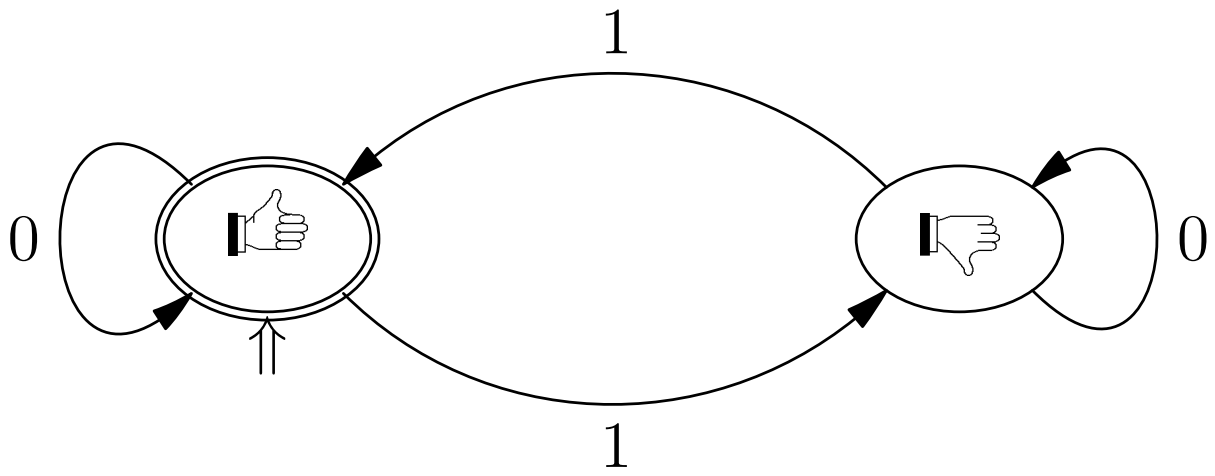
Définir un automate fini $\mathcal{A}$ c'est se donner :

- Un ensemble fini non vide Σ : l'*alphabet* de l'automate
- Un ensemble fini non vide $\mathcal{E}$: les *états* de l'automate
- Un état particulier $I \in \mathcal{E}$: l'état initial de l'automate
- Une partie $A \subset \mathcal{E}$ constituée des *états acceptants*
  Les autres sont les états *refusant*
- Une application $\delta : \mathcal{E} \times \Sigma \to \mathcal{E}$: la fonction de transition de l'automate

Pair/Impair

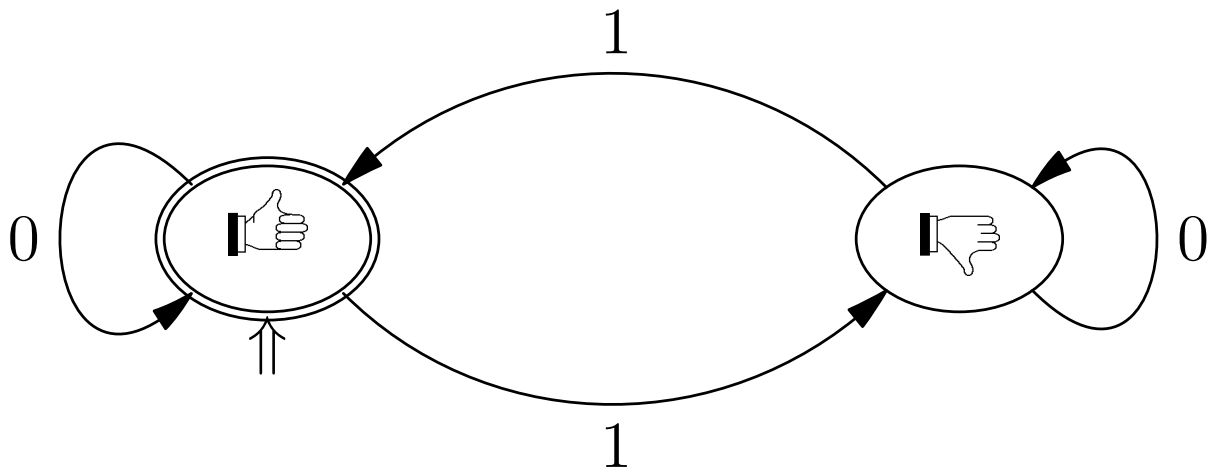


Pair/Impair



Alphabet : $\{0; 1\}$

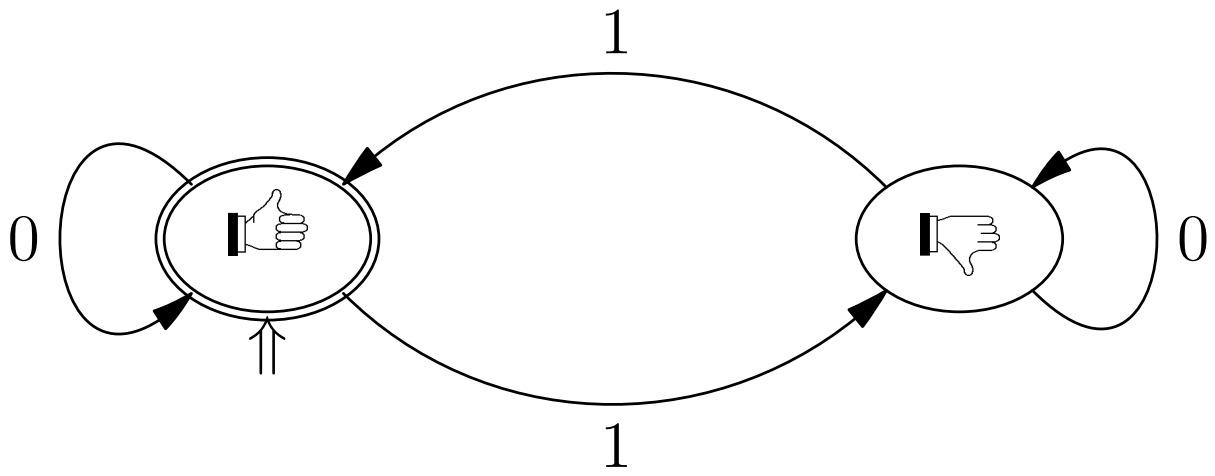
Pair/Impair



Alphabet : $\{0; 1\}$

Ensemble des états : $\mathcal{E} = \{$ 👍 , 👎 $\}$

Pair/Impair

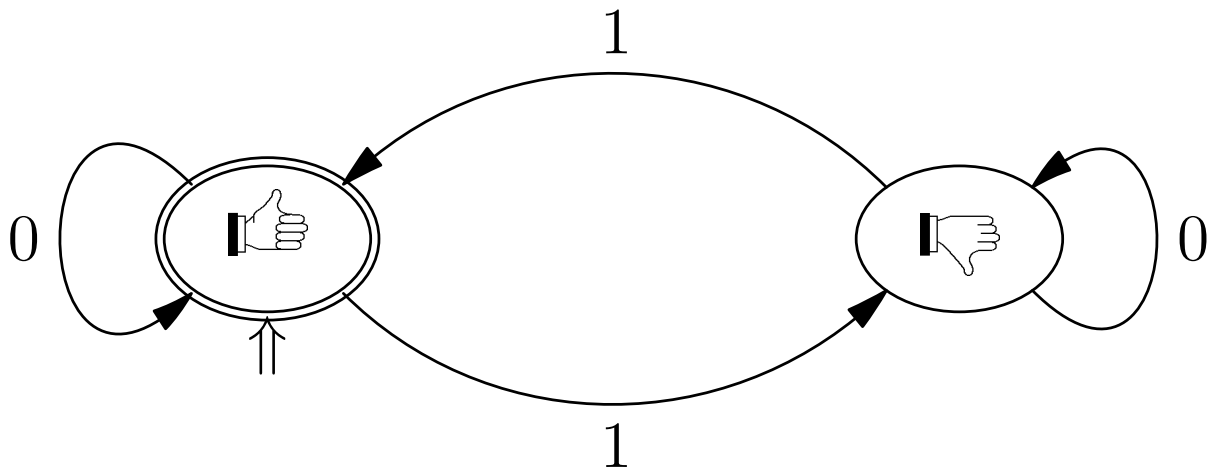


Alphabet : $\{0; 1\}$

Ensemble des états : $\mathcal{E} = \{$ 👍 , 👎 $\}$

État initial : $I =$ 👍

Pair/Impair



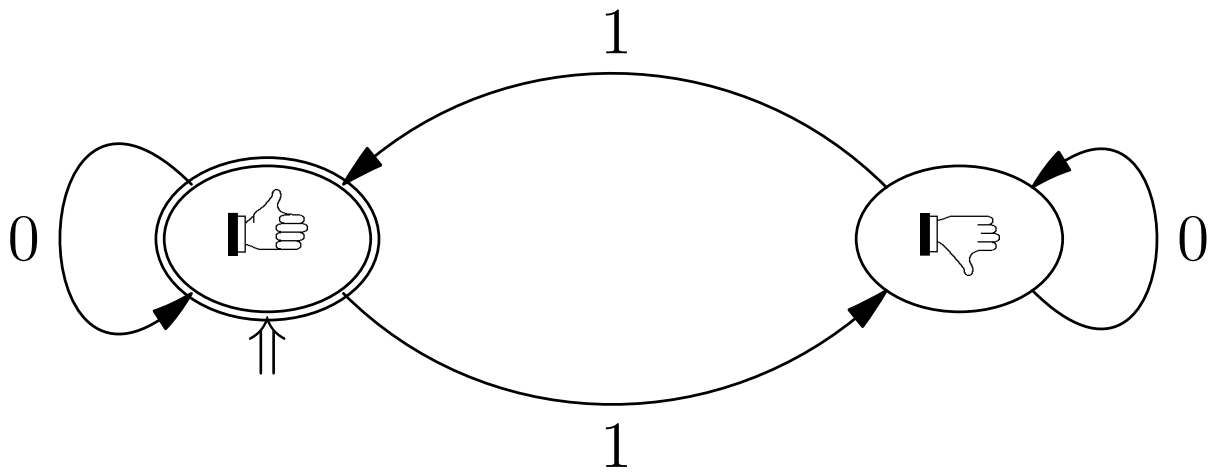
Alphabet : $\{0; 1\}$

Ensemble des états : $\mathcal{E} = \{$ 👍 , 👎 $\}$

État initial : $I =$ 👍

Ensemble des états acceptants : $A = \{$ 👍 $\}$

Pair/Impair



Alphabet : $\{0; 1\}$

Ensemble des états : $\mathcal{E} = \{$ 👍 , 👎 $\}$

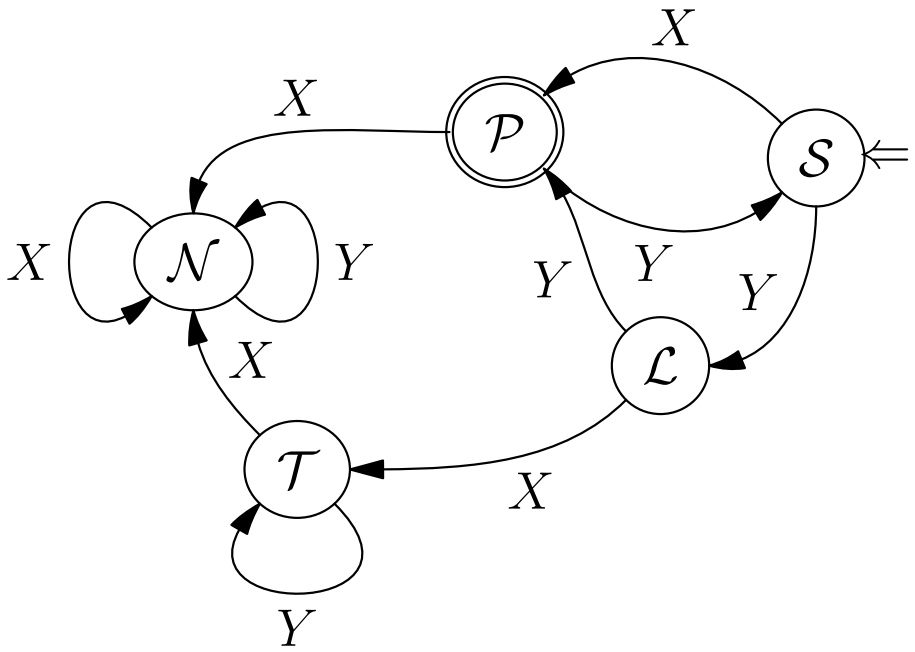
État initial : $I =$ 👍

Ensemble des états acceptants : $A = \{$ 👍 $\}$

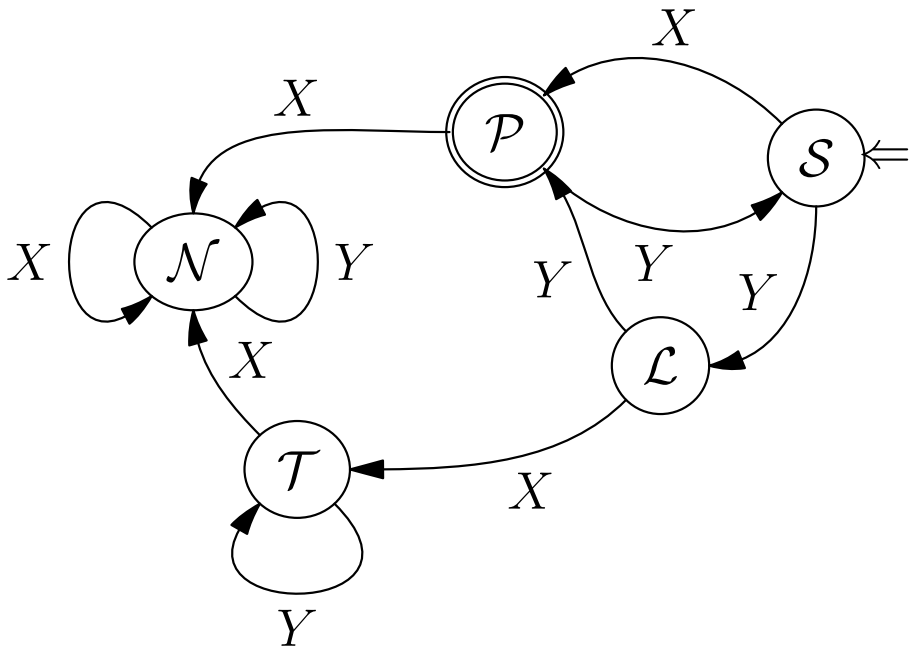
Fonction de transition

	0	1
👍	👍	👎
👎	👎	👎

Jeu

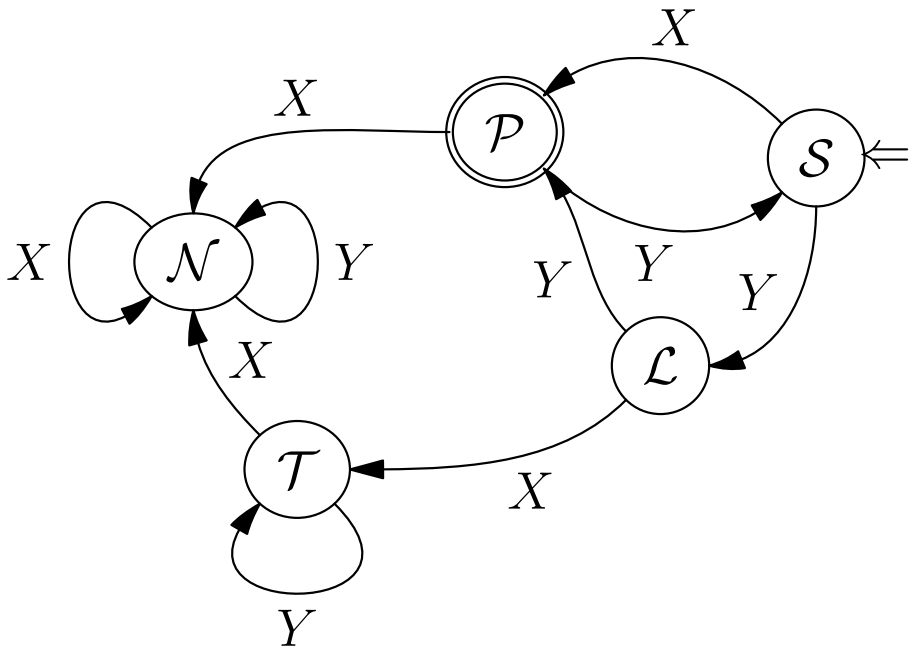

X
X
$\mathcal{P}$
$\mathcal{S}$
X
$\mathcal{N}$
Y
Y
Y
Y
X
$\mathcal{L}$
$\mathcal{T}$
X
Y

Jeu



Alphabet : $\{\mathcal{X}; \mathcal{Y}\}$

Jeu

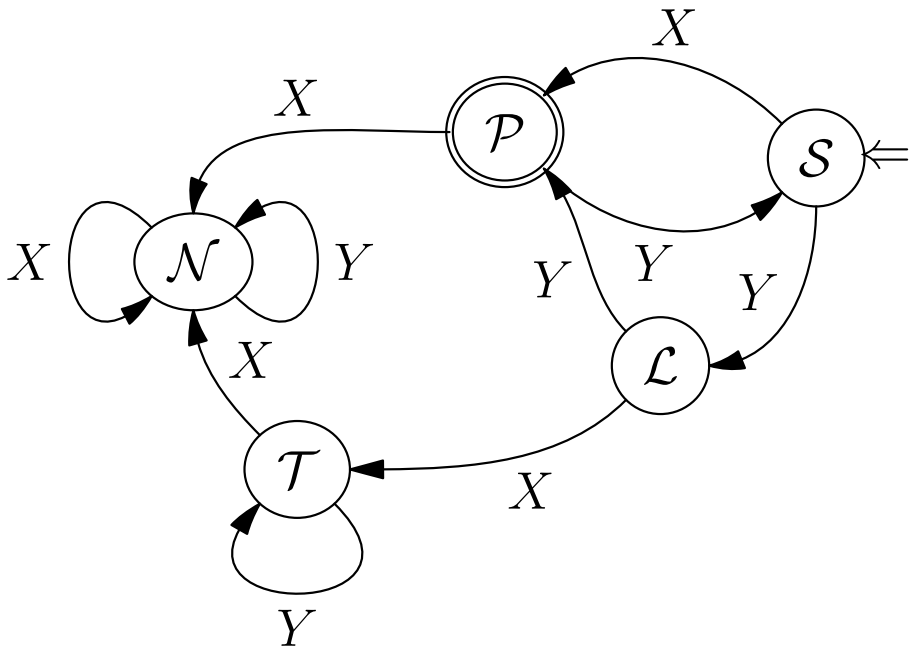


Alphabet : $\{\mathcal{X}; \mathcal{Y}\}$

Ensemble des états :

$$\mathcal{E} = \{L, N, P, S, T\}$$

Jeu



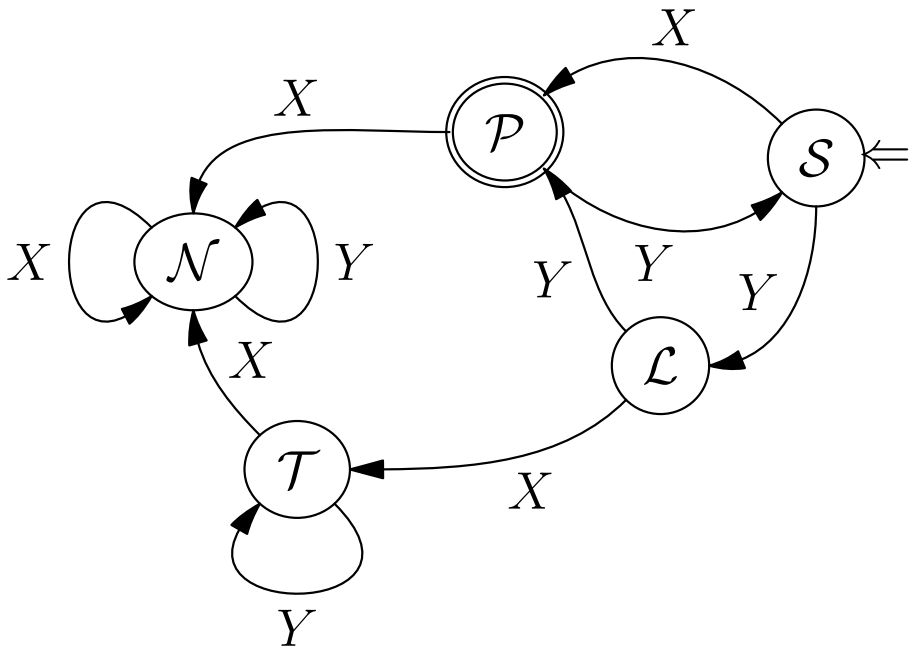
Alphabet : $\{\mathcal{X}; \mathcal{Y}\}$

Ensemble des états :

$$\mathcal{E} = \{L, N, P, S, T\}$$

État initial : $I = S$

Jeu



Alphabet : $\{\mathcal{X}; \mathcal{Y}\}$

Ensemble des états :

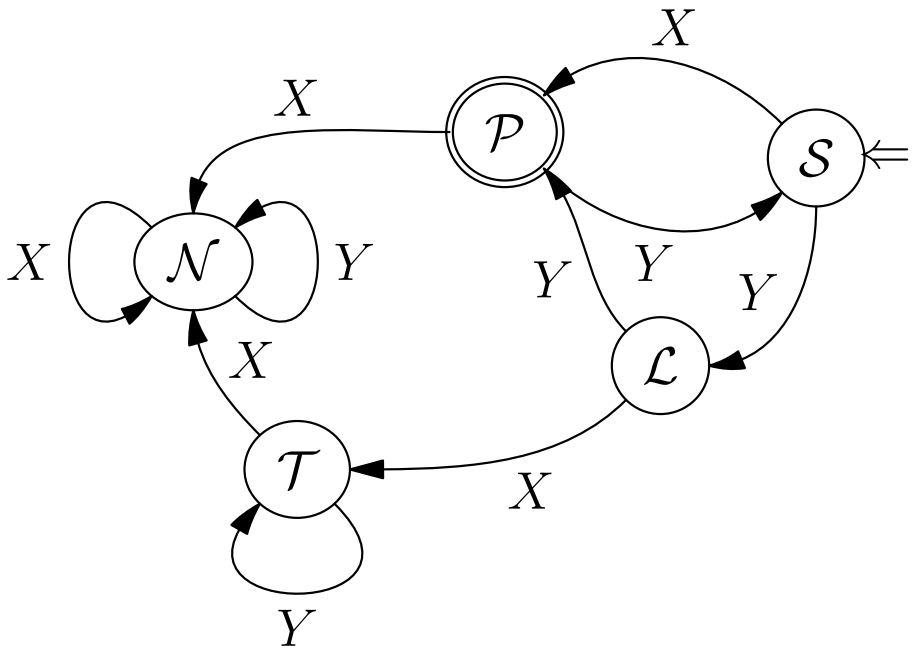
$$\mathcal{E} = \{L, N, P, S, T\}$$

État initial : $I = S$

Ensemble des états acceptants :

$$A = \{P\}$$

Jeu



Alphabet : $\{\mathcal{X};\mathcal{Y}\}$

Ensemble des états :

$$\mathcal{E} = \{L, N, P, S, T\}$$

État initial : $I = S$

Ensemble des états acceptants :

$$A = \{P\}$$

	$\mathcal{X}$	$\mathcal{Y}$
L	T	P
N	N	N
P	N	S
S	P	L
T	N	T

Diagramme de l'automate

Si l'automate n'est pas trop compliqué, on peut construire le *diagramme de l'automate* :

Diagramme de l'automate

Si l'automate n'est pas trop compliqué, on peut construire le *diagramme de l'automate* :
C'est un graphe orienté dont :

Diagramme de l'automate

Si l'automate n'est pas trop compliqué, on peut construire le *diagramme de l'automate* :
C'est un graphe orienté dont :

- les sommets sont les états

Diagramme de l'automate

Si l'automate n'est pas trop compliqué, on peut construire le *diagramme de l'automate* :
C'est un graphe orienté dont :

- les sommets sont les états
- les arêtes représentent les transitions

Diagramme de l'automate

Si l'automate n'est pas trop compliqué, on peut construire le *diagramme de l'automate* :
C'est un graphe orienté dont :

- les sommets sont les états
- les arêtes représentent les transitions

Par convention :

- les états acceptants sont représentés par un cercle double ou sont colorés

Diagramme de l'automate

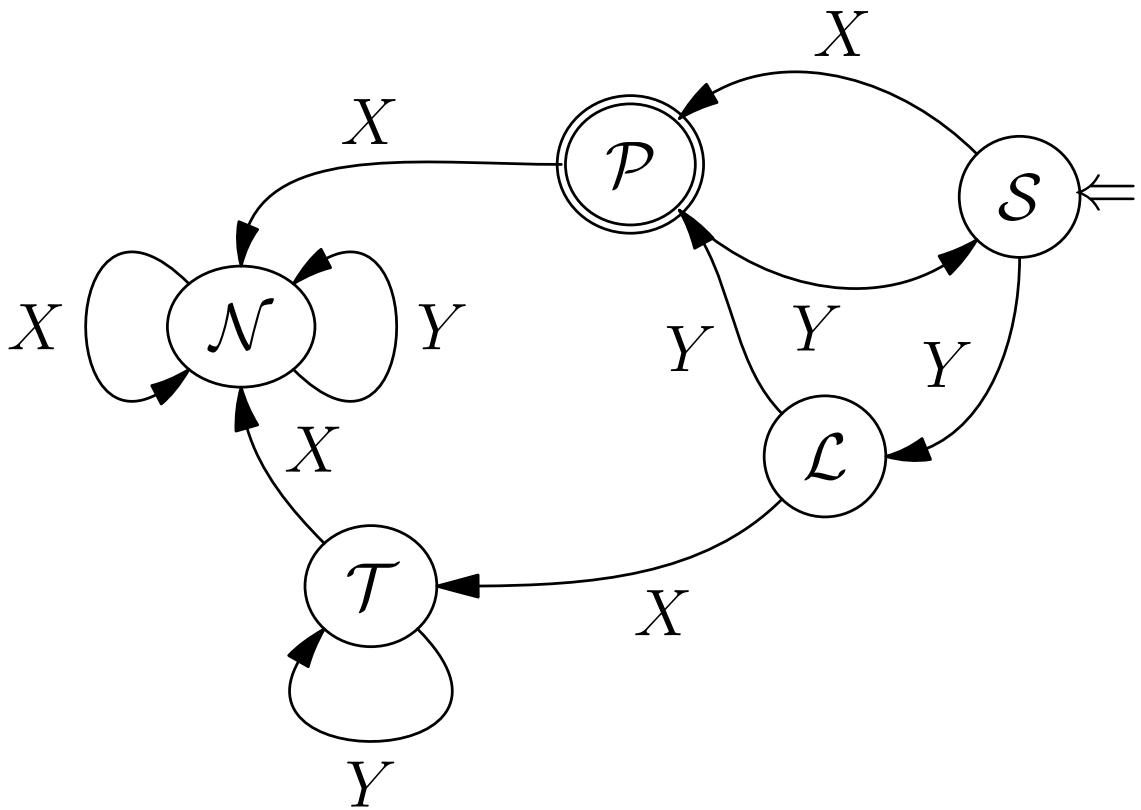
Si l'automate n'est pas trop compliqué, on peut construire le *diagramme de l'automate* :
C'est un graphe orienté dont :

- les sommets sont les états
- les arêtes représentent les transitions

Par convention :

- les états acceptants sont représentés par un cercle double ou sont colorés
- l'état initial est repéré par une grosse flèche $\Rightarrow$

Diagramme de l'automate



Automates finis déterministes complets

Définition

1. Un automate est *déterministe* si :
   - il a un unique état initial ;
   - pour chaque état, il y a au plus une une transition par label

Automates finis déterministes complets

Définition

1. Un automate est *déterministe* si :
   - il a un unique état initial ;
   - pour chaque état, il y a au plus une une transition par label
2. Un automate est *complet* si, pour chaque état, il y a au moins une transition par label.

Automates finis déterministes complets

Définition

1. Un automate est *déterministe* si :
   - il a un unique état initial ;
   - pour chaque état, il y a au plus une une transition par label
2. Un automate est *complet* si, pour chaque état, il y a au moins une transition par label.

Dans toute la suite, tous les automates étudiés seront considérés déterministes et complets

Applications

Reconnaître des mots est une tâche très importante en informatique qu'on retrouve dans des domaines aussi divers que :

Applications

Reconnaître des mots est une tâche très importante en informatique qu'on retrouve dans des domaines aussi divers que :

- les compilateurs qui transforment un code de haut niveau, compréhensible par un humain, en un code machine directement utilisable par l'ordinateur ;

Applications

Reconnaître des mots est une tâche très importante en informatique qu'on retrouve dans des domaines aussi divers que :

- les compilateurs qui transforment un code de haut niveau, compréhensible par un humain, en un code machine directement utilisable par l'ordinateur ;
- les éditeurs de texte et les traitements de texte ;

Applications

Reconnaître des mots est une tâche très importante en informatique qu'on retrouve dans des domaines aussi divers que :

- les compilateurs qui transforment un code de haut niveau, compréhensible par un humain, en un code machine directement utilisable par l'ordinateur ;
- les éditeurs de texte et les traitements de texte ;
- les bases de données, les moteurs de recherche ;

Applications

Reconnaître des mots est une tâche très importante en informatique qu'on retrouve dans des domaines aussi divers que :

- les compilateurs qui transforment un code de haut niveau, compréhensible par un humain, en un code machine directement utilisable par l'ordinateur ;
- les éditeurs de texte et les traitements de texte ;
- les bases de données, les moteurs de recherche ;
- les motifs de pixels sur un écran, . . .

Sommaire

Alphabet — Caractères

Définition

Dans toute la suite :

- Σ désigne un ensemble non vide appelé *alphabet*

Alphabet — Caractères

Définition

Dans toute la suite :

- Σ désigne un ensemble non vide appelé *alphabet*
- les éléments de Σ sont les *caractères*

Alphabet — Caractères

Définition

Dans toute la suite :

- Σ désigne un ensemble non vide appelé *alphabet*
- les éléments de Σ sont les *caractères*
- une suite de longueur n d'éléments de Σ s'appelle une *chaîne de caractères*

Alphabet — Caractères

Définition

Dans toute la suite :

- Σ désigne un ensemble non vide appelé *alphabet*
- les éléments de Σ sont les *caractères*
- une suite de longueur n d'éléments de Σ s'appelle une *chaîne de caractères*

Remarque

1. On appelle souvent les caractères des *lettres* même s'il ne s'agit pas vraiment de lettre comme dans $\Sigma = \mathbb{B} = \{0; 1\}$

Alphabet — Caractères

Définition

Dans toute la suite :

- Σ désigne un ensemble non vide appelé *alphabet*
- les éléments de Σ sont les *caractères*
- une suite de longueur n d'éléments de Σ s'appelle une *chaîne de caractères*

Remarque

1. On appelle souvent les caractères des *lettres* même s'il ne s'agit pas vraiment de lettre comme dans $\Sigma = \mathbb{B} = \{0; 1\}$
2. On dit souvent *mot* à la place de chaîne de caractères

Mot sans lettre

Définition (mot sans lettre)

On note ε un mot particulier qui a un nom mais ne possède pas de caractère.

Ce mot est le *mot sans lettre*

Mot sans lettre

Définition (mot sans lettre)

On note ε un mot particulier qui a un nom mais ne possède pas de caractère.
Ce mot est le *mot sans lettre*

- Le mot sans lettre a pour longueur 0

Mot sans lettre

Définition (mot sans lettre)

On note ε un mot particulier qui a un nom mais ne possède pas de caractère.
Ce mot est le *mot sans lettre*

- Le mot sans lettre a pour longueur 0
- C'est la seule chaine de caractère qui a une longueur nulle

Langages

Définition

Soit Σ un alphabet. On note :

Langages

Définition

Soit Σ un alphabet. On note :

1. Σ^n l'ensemble de tous les mots de longueur n ;

Langages

Définition

Soit Σ un alphabet. On note :

1. Σ^n l'ensemble de tous les mots de longueur n ;
2. Σ^* l'ensemble de tous les mots y compris le mot sans lettre

Langages

Définition

Soit Σ un alphabet. On note :

1. Σ^n l'ensemble de tous les mots de longueur n ;
2. Σ^* l'ensemble de tous les mots y compris le mot sans lettre

Exemple

Pour $\Sigma = \{\uparrow\}$ cela donne :

Langages

Définition

Soit Σ un alphabet. On note :

1. Σ^n l'ensemble de tous les mots de longueur n ;
2. Σ^* l'ensemble de tous les mots y compris le mot sans lettre

Exemple

Pour $\Sigma = \{\uparrow\}$ cela donne :

- $\Sigma^0 = \{\varepsilon\}$;

Langages

Définition

Soit Σ un alphabet. On note :

1. Σ^n l'ensemble de tous les mots de longueur n ;
2. Σ^* l'ensemble de tous les mots y compris le mot sans lettre

Exemple

Pour $\Sigma = \{\uparrow\}$ cela donne :

- $\Sigma^0 = \{\varepsilon\}$;
- $\Sigma^1 = \{\uparrow\}$;

Langages

Définition

Soit Σ un alphabet. On note :

1. Σ^n l'ensemble de tous les mots de longueur n ;
2. Σ^* l'ensemble de tous les mots y compris le mot sans lettre

Exemple

Pour $\Sigma = \{\uparrow\}$ cela donne :

- $\Sigma^0 = \{\varepsilon\}$;
- $\Sigma^1 = \{\uparrow\}$;
- $\Sigma^2 = \{\uparrow\uparrow\}$;

Langages

Définition

Soit Σ un alphabet. On note :

1. Σ^n l'ensemble de tous les mots de longueur n ;
2. Σ^* l'ensemble de tous les mots y compris le mot sans lettre

Exemple

Pour $\Sigma = \{\uparrow\}$ cela donne :

- $\Sigma^0 = \{\varepsilon\}$;
- $\Sigma^1 = \{\uparrow\}$;
- $\Sigma^2 = \{\uparrow\uparrow\}$;
- $\Sigma^3 = \{\uparrow\uparrow\uparrow\}$, etc.

Langages

Définition

Soit Σ un alphabet. On note :

1. Σ^n l'ensemble de tous les mots de longueur n ;
2. Σ^* l'ensemble de tous les mots y compris le mot sans lettre

Exemple

Pour $\Sigma = \{\uparrow\}$ cela donne :

- $\Sigma^0 = \{\varepsilon\}$;
- $\Sigma^1 = \{\uparrow\}$;
- $\Sigma^2 = \{\uparrow\uparrow\}$;
- $\Sigma^3 = \{\uparrow\uparrow\uparrow\}$, etc.
- $\Sigma^* = \{\varepsilon, \uparrow, \uparrow\uparrow, \uparrow\uparrow\uparrow, \uparrow\uparrow\uparrow\uparrow, \uparrow\uparrow\uparrow\uparrow\uparrow, \uparrow\uparrow\uparrow\uparrow\uparrow\uparrow, \uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow, \ldots\}$

Langages

Exemple

Si $\Sigma = \mathbb{B} = \{0; 1\}$ alors :

Langages

Exemple

Si $\Sigma = \mathbb{B} = \{0; 1\}$ alors :

- $\Sigma^0 = \{\varepsilon\}$;

Langages

Exemple

Si $\Sigma = \mathbb{B} = \{0; 1\}$ alors :

- $\Sigma^0 = \{\varepsilon\}$;
- $\Sigma^1 = \{0; 1\}$;

Langages

Exemple

Si $\Sigma = \mathbb{B} = \{0; 1\}$ alors :

- $\Sigma^0 = \{\varepsilon\}$;
- $\Sigma^1 = \{0; 1\}$;
- $\Sigma^2 = \{00; 01; 10; 11\}$;

Langages

Exemple

Si $\Sigma = \mathbb{B} = \{0; 1\}$ alors :

- $\Sigma^0 = \{\varepsilon\}$;
- $\Sigma^1 = \{0; 1\}$;
- $\Sigma^2 = \{00; 01; 10; 11\}$;
- $\Sigma^3 = \{000; 001; 010; 011; 100; 101; 110; 111\}$, etc.

Langages

Exemple

Si $\Sigma = \mathbb{B} = \{0; 1\}$ alors :

- $\Sigma^0 = \{\varepsilon\}$;
- $\Sigma^1 = \{0; 1\}$;
- $\Sigma^2 = \{00; 01; 10; 11\}$;
- $\Sigma^3 = \{000; 001; 010; 011; 100; 101; 110; 111\}$, etc.

$$\Sigma^* = \{\varepsilon; \underbrace{0; 1}_{\Sigma^1}; \underbrace{00; 01; 10; 11}_{\Sigma^2}; \underbrace{000; 001; 010; 011; 100; 101; 110; 111}_{\Sigma^3}; \ldots\}$$

Langages

Définition

Un *langage* est un sous-ensemble de Σ^*.

Langages

Définition

Un *langage* est un sous-ensemble de Σ^*.

Définition

On dit que la langage M est *plus grand* que le langage L si :

$$L \subset M$$

Langages

Définition

Un *langage* est un sous-ensemble de Σ^*.

Définition

On dit que la langage M est *plus grand* que le langage L si :

$$L \subset M$$

Remarque

1. Un langage est simplement un ensemble de mot ;

Langages

Définition

Un *langage* est un sous-ensemble de Σ^*.

Définition

On dit que la langage M est *plus grand* que le langage L si :

$$L \subset M$$

Remarque

1. Un langage est simplement un ensemble de mot ;
2. L'ensemble vide $\emptyset$ est le plus petit langage construit avec les éléments de Σ ; c'est le *langage vide* ;

Langages

Définition

Un *langage* est un sous-ensemble de Σ^*.

Définition

On dit que la langage M est *plus grand* que le langage L si :

$$L \subset M$$

Remarque

1. Un langage est simplement un ensemble de mot ;
2. L'ensemble vide $\emptyset$ est le plus petit langage construit avec les éléments de Σ ; c'est le *langage vide* ;
3. Σ^* est le plus grand langage.

Langages

Exemple

Voici quelques langages lorsque $\Sigma = \{a; b\}$:

Langages

Exemple

Voici quelques langages lorsque $\Sigma = \{a; b\}$:

1. Le langage $\{\varepsilon\}$ réduit au mot sans lettre ;

Langages

Exemple

Voici quelques langages lorsque $\Sigma = \{a; b\}$:

1. Le langage $\{\varepsilon\}$ réduit au mot sans lettre ;
   À ne pas confondre avec la langage vide !

Langages

Exemple

Voici quelques langages lorsque $\Sigma = \{a; b\}$:

1. Le langage $\{\varepsilon\}$ réduit au mot sans lettre ;
   À ne pas confondre avec la langage vide !
2. le langage des mots qui contiennent au moins deux fois le caractère a ;

Langages

Exemple

Voici quelques langages lorsque $\Sigma = \{a; b\}$:

1. Le langage $\{\varepsilon\}$ réduit au mot sans lettre ;
   À ne pas confondre avec la langage vide !
2. le langage des mots qui contiennent au moins deux fois le caractère a ;
3. le langage des mots qui autant de a que de b.

Sommaire

Somme

Dans ce qui suit, les langages sont tous construits à partir du même alphabet Σ.

Somme

Dans ce qui suit, les langages sont tous construits à partir du même alphabet Σ.

Puisque les langages sont des parties de Σ^*, on peut utiliser les opérations ensemblistes :

réunion $\cup$; intersection $\cap$; complémentaire ^{c}

Somme

Dans ce qui suit, les langages sont tous construits à partir du même alphabet Σ.

Puisque les langages sont des parties de Σ^*, on peut utiliser les opérations ensemblistes :

$$\text{réunion } \cup \text{ ; intersection } \cap \text{ ; complémentaire } {}^c$$

Parmi ces trois opérations, la réunion est celle qui joue le plus grand rôle ;

Somme

Pour éviter d'utiliser le symbole $\cup$, on note

$$L_1 + L_2$$

la réunion des deux langages L_1 et L_2.

Somme

Pour éviter d'utiliser le symbole $\cup$, on note

$$L_1 + L_2$$

la réunion des deux langages L_1 et L_2.
On appelle la *somme* de L_1 et L_2 le nouveau langage obtenu.

Somme

Cette addition des langages possède les propriétés de la réunion ensembliste :

Somme

Cette addition des langages possède les propriétés de la réunion ensembliste :

1. $L_1 + L_2 = L_2 + L_1$

Somme

Cette addition des langages possède les propriétés de la réunion ensembliste :

1. $L_1 + L_2 = L_2 + L_1$
2. $(L_1 + L_2) + L_3 = L_1 + (L_2 + L_3) = L_1 + L_2 + L_3$

Somme

Cette addition des langages possède les propriétés de la réunion ensembliste :

1. $L_1 + L_2 = L_2 + L_1$
2. $(L_1 + L_2) + L_3 = L_1 + (L_2 + L_3) = L_1 + L_2 + L_3$
3. $L + L = L$

Somme

Cette addition des langages possède les propriétés de la réunion ensembliste :

1. $L_1 + L_2 = L_2 + L_1$
2. $(L_1 + L_2) + L_3 = L_1 + (L_2 + L_3) = L_1 + L_2 + L_3$
3. $L + L = L$
4. $L + \emptyset = \emptyset + L = L$

Somme

Cette addition des langages possède les propriétés de la réunion ensembliste :

1. $L_1 + L_2 = L_2 + L_1$
2. $(L_1 + L_2) + L_3 = L_1 + (L_2 + L_3) = L_1 + L_2 + L_3$
3. $L + L = L$
4. $L + \emptyset = \emptyset + L = L$
5. $L + \Sigma^* = \Sigma^* = \Sigma^*$

Somme

Cette addition des langages possède les propriétés de la réunion ensembliste :

1. $L_1 + L_2 = L_2 + L_1$
2. $(L_1 + L_2) + L_3 = L_1 + (L_2 + L_3) = L_1 + L_2 + L_3$
3. $L + L = L$
4. $L + \emptyset = \emptyset + L = L$
5. $L + \Sigma^* = \Sigma^* = \Sigma^*$
6. $L_1 + L_2 = L_2 \iff L_1 \subset L_2$

Somme

Remarque

1. L'ensemble vide $\emptyset$ joue le rôle du 0 pour l'addition des langages ;

Somme

Remarque

1. L'ensemble vide $\emptyset$ joue le rôle du 0 pour l'addition des langages ;
2. L'équation ensembliste

$$L + X = M$$

peut avoir plusieurs solutions

Somme

Remarque

1. L'ensemble vide $\emptyset$ joue le rôle du 0 pour l'addition des langages ;
2. L'équation ensembliste

$$L + X = M$$

peut avoir plusieurs solutions
ce qui empêche d'avoir une soustraction !

Produit de concaténation

Soit σ et τ deux mots d'un langage L.

Le *produit de concaténation* de σ et τ est le mot obtenu en écrivant les caractères de τ à la suite des ceux de σ.

Produit de concaténation

Soit σ et τ deux mots d'un langage L.

Le *produit de concaténation* de σ et τ est le mot obtenu en écrivant les caractères de τ à la suite des ceux de σ.

On le note $\sigma \cdot \tau$ ou plus simplement $\sigma\tau$.

Produit de concaténation

Soit σ et τ deux mots d'un langage L.

Le *produit de concaténation* de σ et τ est le mot obtenu en écrivant les caractères de τ à la suite des ceux de σ.

On le note $\sigma \cdot \tau$ ou plus simplement $\sigma\tau$.

Exemple

Soit $\Sigma = \{a; b; c\}$. Avec $\sigma = aabc$ et $\tau = aca$, on obtient le mot :

$$\sigma\tau = aabcaca$$

Produit de concaténation

Définition

Soit σ un mot de longueur $n > 0$ et τ un mot de longueur $m > 0$.
Leur concaténation $\nu = \sigma\tau$ est le mot de longueur $m+n$ défini par :

$$\begin{aligned} \nu(k) &= \sigma(k) && \text{si } 1 \leqslant k \leqslant n \\ \nu(k) &= \tau(k-n) && \text{si } n < k \leqslant m \end{aligned}$$

où $\nu(k)$ désigne le k^{e} caractère du mot ν.

Produit de concaténation

Définition

Soit σ un mot de longueur $n > 0$ et τ un mot de longueur $m > 0$.
Leur concaténation $\nu = \sigma\tau$ est le mot de longueur $m+n$ défini par :

$$\begin{aligned} \nu(k) &= \sigma(k) && \text{si } 1 \leqslant k \leqslant n \\ \nu(k) &= \tau(k-n) && \text{si } n < k \leqslant m \end{aligned}$$

où $\nu(k)$ désigne le k^{e} caractère du mot ν.

Exemple

Avec $\sigma = aabc$ et $\tau = aca$, on obtient le mot :

$$\sigma\tau = aabcaca$$

qui est bien de longueur 7.

Produit de concaténation

Propriété

1. *Pour tout mot* σ, *on a* $\sigma \cdot \varepsilon = \varepsilon \cdot \sigma = \sigma$

Produit de concaténation

Propriété

1. *Pour tout mot* σ, *on a* $\sigma \cdot \varepsilon = \varepsilon \cdot \sigma = \sigma$
   le mot ε *joue le même rôle que 1 pour la multiplication des nombres*

Produit de concaténation

Propriété

1. *Pour tout mot* σ, *on a* $\sigma \cdot \varepsilon = \varepsilon \cdot \sigma = \sigma$
   le mot ε *joue le même rôle que 1 pour la multiplication des nombres*
2. *La concaténation est associative :*

$$\sigma(\tau\nu) = (\sigma\tau)\nu$$

Produit de concaténation

Propriété

1. *Pour tout mot* σ, *on a* $\sigma \cdot \varepsilon = \varepsilon \cdot \sigma = \sigma$
   le mot ε *joue le même rôle que 1 pour la multiplication des nombres*
2. *La concaténation est associative :*

$$\sigma(\tau\nu) = (\sigma\tau)\nu$$

3. *Par contre, la concaténation n'est pas commutative. En général :*

$$\sigma\tau \neq \tau\sigma$$

Produit de concaténation

Propriété (Notations)

On pose :

- $\sigma^0 = \varepsilon$

Produit de concaténation

Propriété (Notations)

On pose :

- $\sigma^0 = \varepsilon$
- $\sigma^1 = \sigma$

Produit de concaténation

Propriété (Notations)

On pose :

- $\sigma^0 = \varepsilon$
- $\sigma^1 = \sigma$
- $\sigma^2 = \sigma \cdot \sigma$

Produit de concaténation

Propriété (Notations)

On pose :

- $\sigma^0 = \varepsilon$
- $\sigma^1 = \sigma$
- $\sigma^2 = \sigma \cdot \sigma$
- $\sigma^3 = \sigma \cdot \sigma^2$, *etc.*

Produit de concaténation

Propriété (Notations)

On pose :

- $\sigma^0 = \varepsilon$
- $\sigma^1 = \sigma$
- $\sigma^2 = \sigma \cdot \sigma$
- $\sigma^3 = \sigma \cdot \sigma^2$, *etc.*

On obtient la formule pour tout n entier naturel :

$$\sigma^m \cdot \sigma^n = \sigma^{n+m}$$

Produit de concaténation

Propriété (Notations)

On pose :

- $\sigma^0 = \varepsilon$
- $\sigma^1 = \sigma$
- $\sigma^2 = \sigma \cdot \sigma$
- $\sigma^3 = \sigma \cdot \sigma^2$, *etc.*

On obtient la formule pour tout n entier naturel :

$$\sigma^m \cdot \sigma^n = \sigma^{n+m}$$

Exemple

le mot *aaabbbabbb* peut tout aussi bien s'écrire $a^3b^2ab^3$.

Produit de concaténation

Généralisons l'opération de concaténation aux langages :

Définition

Si L_1 et L_2 sont deux langages, la *concaténation* de L_1 et L_2 est le langage :

$$L_1 \cdot L_2 = \{\sigma_1\sigma_2 \mid \sigma_1 \in L_1 \text{ et } \sigma_2 \in L_2\}$$

Produit de concaténation

Généralisons l'opération de concaténation aux langages :

Définition

Si L_1 et L_2 sont deux langages, la *concaténation* de L_1 et L_2 est le langage :

$$L_1 \cdot L_2 = \{\sigma_1\sigma_2 \mid \sigma_1 \in L_1 \text{ et } \sigma_2 \in L_2\}$$

Exemple

Si $L_1 = \{\varepsilon, b\}$ et $L_2 = \{a, ba\}$ alors :

$$L_1 \cdot L_2 = \{a, ba, b^2a\} \quad \text{et} \quad L_2 \cdot L_1 = \{a, ab, ba, bab\}$$

Produit de concaténation

Généralisons l'opération de concaténation aux langages :

Définition

Si L_1 et L_2 sont deux langages, la *concaténation* de L_1 et L_2 est le langage :

$$L_1 \cdot L_2 = \{\sigma_1\sigma_2 \mid \sigma_1 \in L_1 \text{ et } \sigma_2 \in L_2\}$$

Exemple

Si $L_1 = \{\varepsilon, b\}$ et $L_2 = \{a, ba\}$ alors :

$$L_1 \cdot L_2 = \{a, ba, b^2a\} \quad \text{et} \quad L_2 \cdot L_1 = \{a, ab, ba, bab\}$$

On a donc $L_1 \cdot L_2 \neq L_2 \cdot L_1$

Produit de concaténation

Exemple

Le langage $\Sigma^* a \Sigma^*$ est constitué des mots $\sigma = \sigma_1 a \sigma_2$ avec σ_1 et σ_2 quelconques.

Produit de concaténation

Exemple

Le langage $\Sigma^* a \Sigma^*$ est constitué des mots $\sigma = \sigma_1 a \sigma_2$ avec σ_1 et σ_2 quelconques.
Le langage $\Sigma^* a \Sigma^*$ est ainsi le langage des mots qui contiennent le caractère a

Produit de concaténation

Propriété

1. *Le produit de concaténation est doublement distributif par rapport à l'addition des langages :*

$$M(L_1 + L_2) = ML_1 + ML_2 \quad et \quad (L_1 + L_2)M = L_1M + L_2M$$

Produit de concaténation

Propriété

1. *Le produit de concaténation est doublement distributif par rapport à l'addition des langages :*

$$M(L_1 + L_2) = ML_1 + ML_2 \quad et \quad (L_1 + L_2)M = L_1M + L_2M$$

2. *Rôle de $\emptyset$ et ε :*

$$L\emptyset = \emptyset L = \emptyset \quad et \quad L\varepsilon = \varepsilon L = L$$

Produit de concaténation

Comme pour les caractères, on peut définir la puissance d'un langage :

Définition

1. $L^0 = \varepsilon$ et $L^{n+1} = L \cdot L^n$

Produit de concaténation

Comme pour les caractères, on peut définir la puissance d'un langage :

Définition

1. $L^0 = \varepsilon$ et $L^{n+1} = L \cdot L^n$
2. On appelle *étoile* de L le langage :

$$L^* = L^0 + L^1 + L^2 + \cdots + L^n + \cdots$$

Produit de concaténation

Comme pour les caractères, on peut définir la puissance d'un langage :

Définition

1. $L^0 = \varepsilon$ et $L^{n+1} = L \cdot L^n$
2. On appelle *étoile* de L le langage :

$$L^* = L^0 + L^1 + L^2 + \cdots + L^n + \cdots$$

3. On introduit aussi :

$$L^+ = L^1 + L^2 + \cdots + L^n + \cdots$$

Produit de concaténation

Comme pour les caractères, on peut définir la puissance d'un langage :

Définition

1. $L^0 = \varepsilon$ et $L^{n+1} = L \cdot L^n$
2. On appelle *étoile* de L le langage :

$$L^* = L^0 + L^1 + L^2 + \cdots + L^n + \cdots$$

3. On introduit aussi :

$$L^+ = L^1 + L^2 + \cdots + L^n + \cdots$$

Si $\varepsilon \in L$ il n'y a pas de différence entre L^+ et L^*.
Sinon, $L^* = \varepsilon + L^+$.

Produit de concaténation

Propriété

On a :

$$L^+ = L^*L = LL^*$$
$$L^* = \varepsilon + L^+$$
$$L^* = \varepsilon + LL^* + \varepsilon + L^*L$$

Sommaire

Problématique

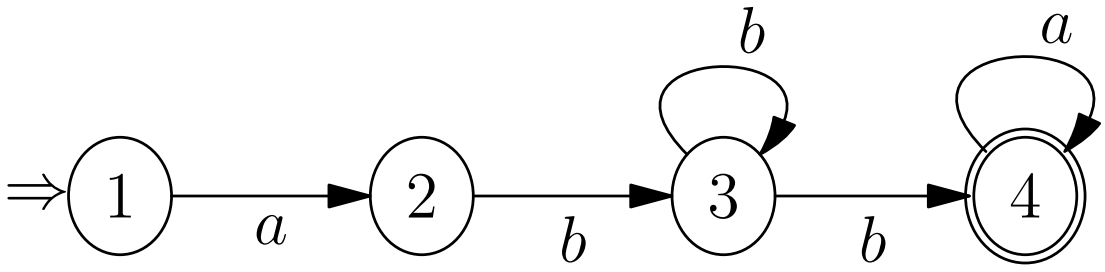
Soit Σ l'alphabet d'un automate fini.
Les différents états sont notés 1, 2, ..., n
L est le langage de cet automate

Problématique

Soit Σ l'alphabet d'un automate fini.
Les différents états sont notés 1, 2, ..., n
L est le langage de cet automate

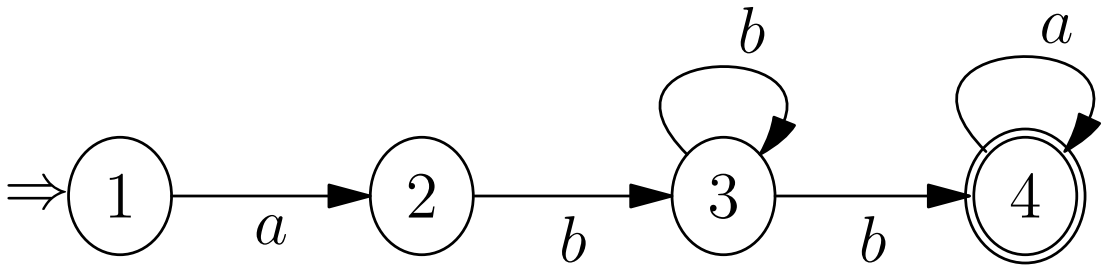
Problématique : Comment caractériser L ?

Exemple



Si le graphe est simple, l'observation de l'automate peut suffire :

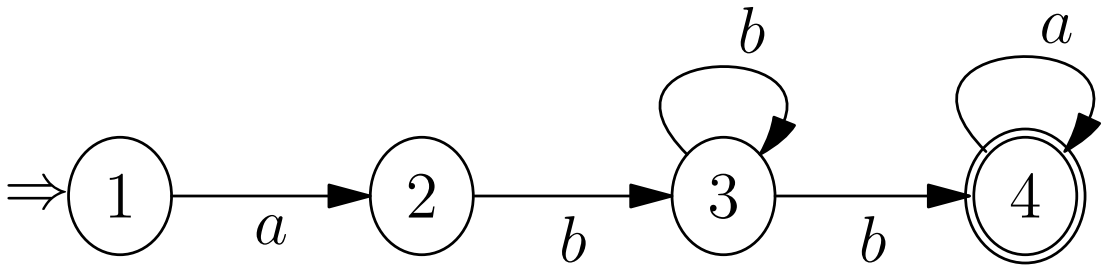
Exemple



Si le graphe est simple, l'observation de l'automate peut suffire :

Le langage engendré par l'automate est abb^*ba^*

Exemple



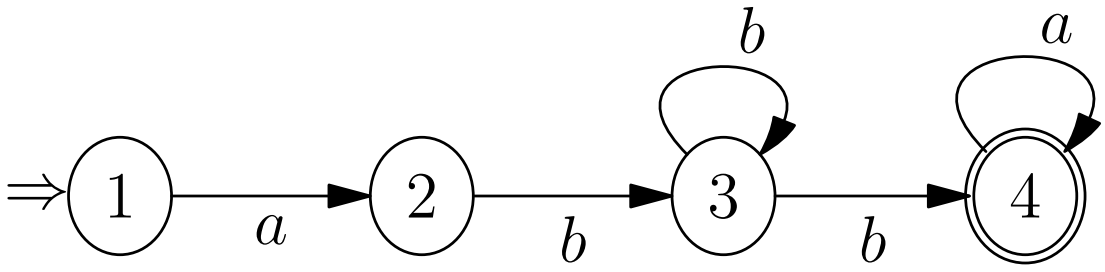
Si le graphe est simple, l'observation de l'automate peut suffire :

Le langage engendré par l'automate est abb^*ba^*

Remarque (Notations)

Attention aux abus de langage !
Ici, on parle de langage, a^* et b^* représente l'étoile du langage engendré par les alphabets $\{a\}$ et $\{b\}$.

Exemple



Si le graphe est simple, l'observation de l'automate peut suffire :

Le langage engendré par l'automate est abb^*ba^*

Remarque (Notations)

Attention aux abus de langage !
Ici, on parle de langage, a^* et b^* représente l'étoile du langage engendré par les alphabets $\{a\}$ et $\{b\}$.
Il s'agit des mots commençant par ab, suivi d'un nombre quelconque de b puis d'un b puis d'un nombre quelconque de a

Langage d'un automate

Dans le cas d'un automate plus complexe, la recherche du langage d'un automate est un problème difficile

Nous étudierons une méthode :

Langage d'un automate

Dans le cas d'un automate plus complexe, la recherche du langage d'un automate est un problème difficile

Nous étudierons une méthode :

- la méthode du départ ;

Langage d'un automate

Dans le cas d'un automate plus complexe, la recherche du langage d'un automate est un problème difficile

Nous étudierons une méthode :

- la méthode du départ ;
- il existe aussi la méthode de l'arrivée

Méthode du départ

Pour chaque état k, on note D_k le langage qui serait accepté par l'automate si k était l'état initial

Méthode du départ

Pour chaque état k, on note D_k le langage qui serait accepté par l'automate si k était l'état initial

Les langages D_1, D_2, ..., D_n sont liés entre eux par des équations

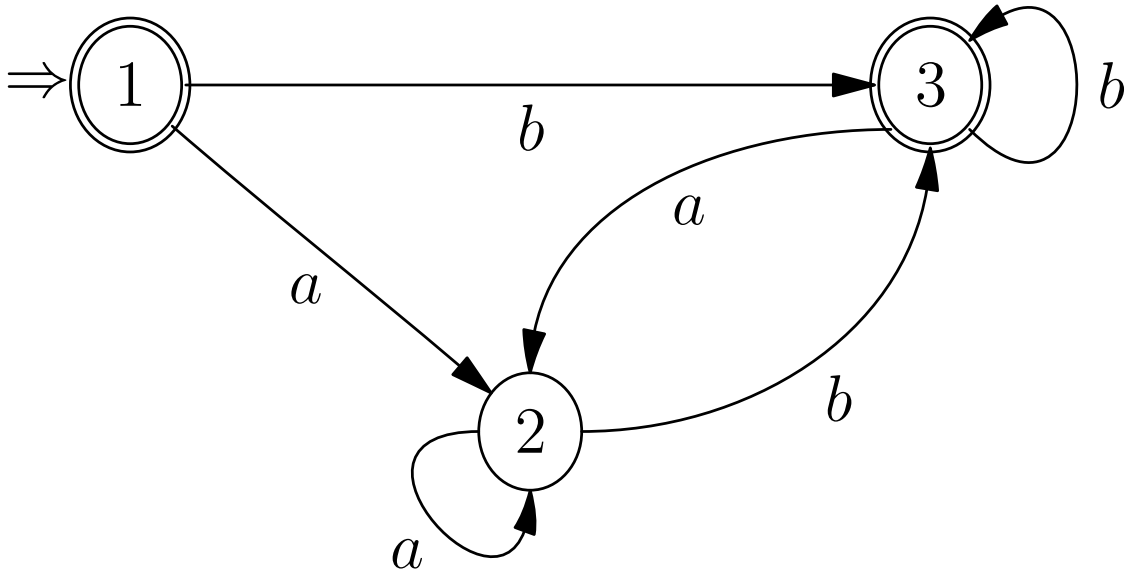
Méthode du départ

Pour chaque état k, on note D_k le langage qui serait accepté par l'automate si k était l'état initial

Les langages D_1, D_2, ..., D_n sont liés entre eux par des équations

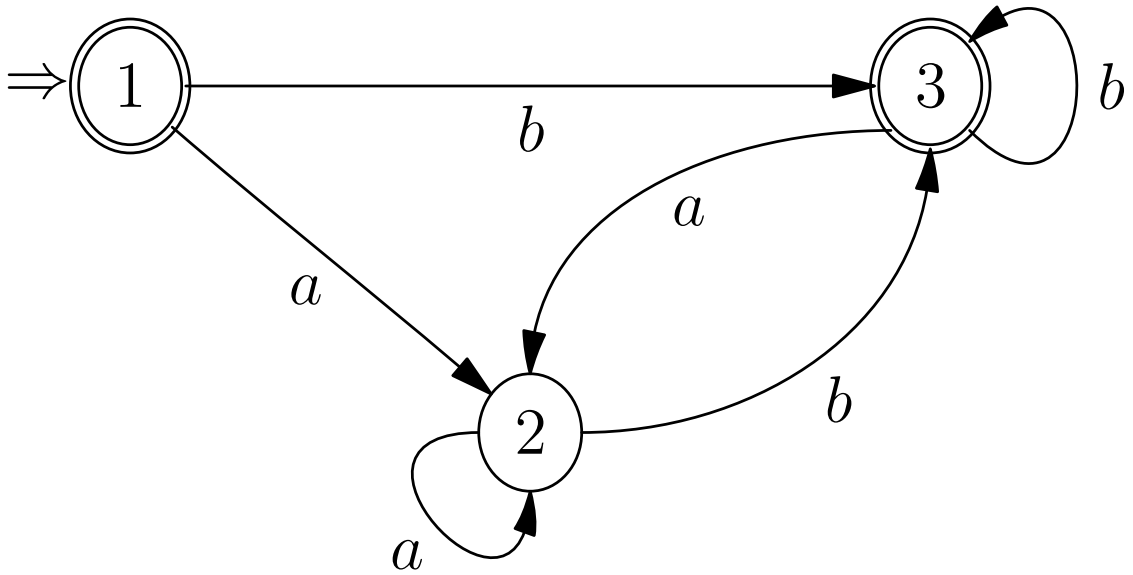
Regardons un Ex

Méthode du départ



On a :

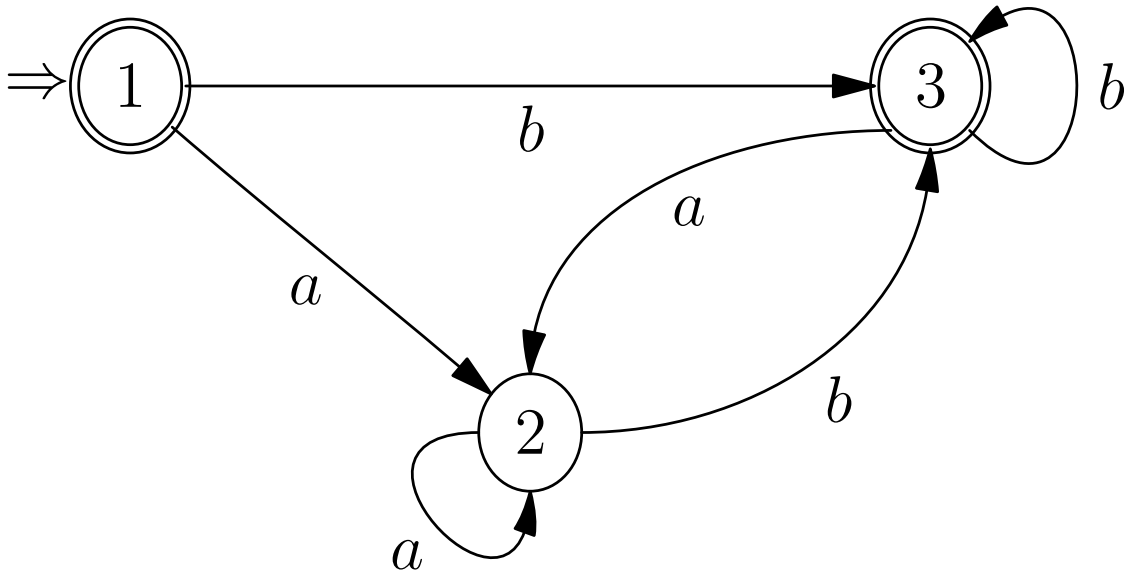
Méthode du départ



On a :

- $D_1 = \varepsilon + aD_2 + bD_3$

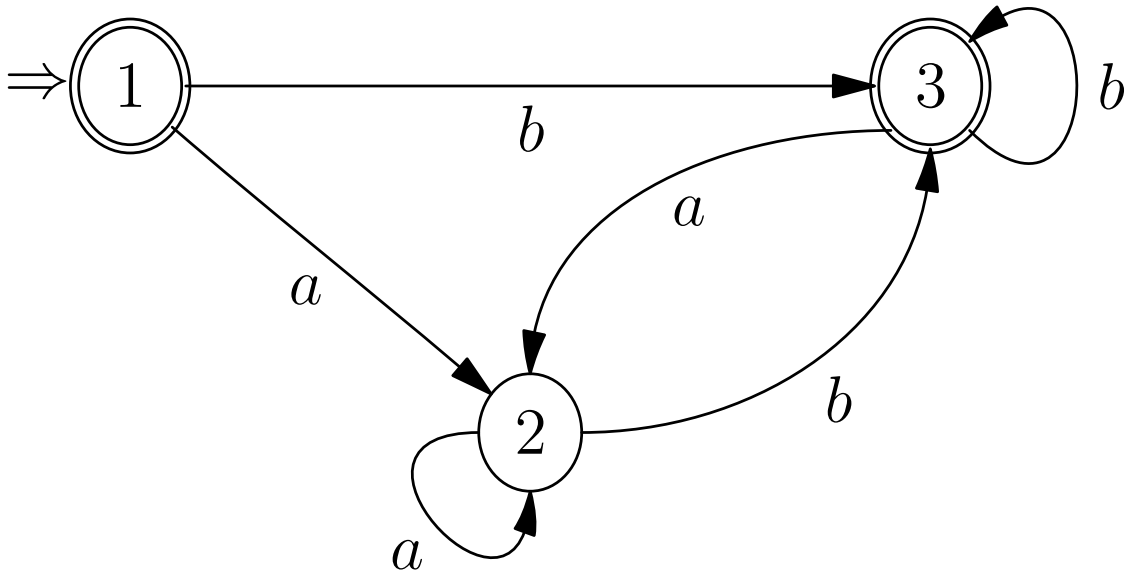
Méthode du départ



On a :

- $D_1 = \varepsilon + aD_2 + bD_3$
- $D_2 = aD_2 + bD_3$

Méthode du départ



On a :

- $D_1 = \varepsilon + aD_2 + bD_3$
- $D_2 = aD_2 + bD_3$
- $D_3 = \varepsilon + aD_2 + bD_3$

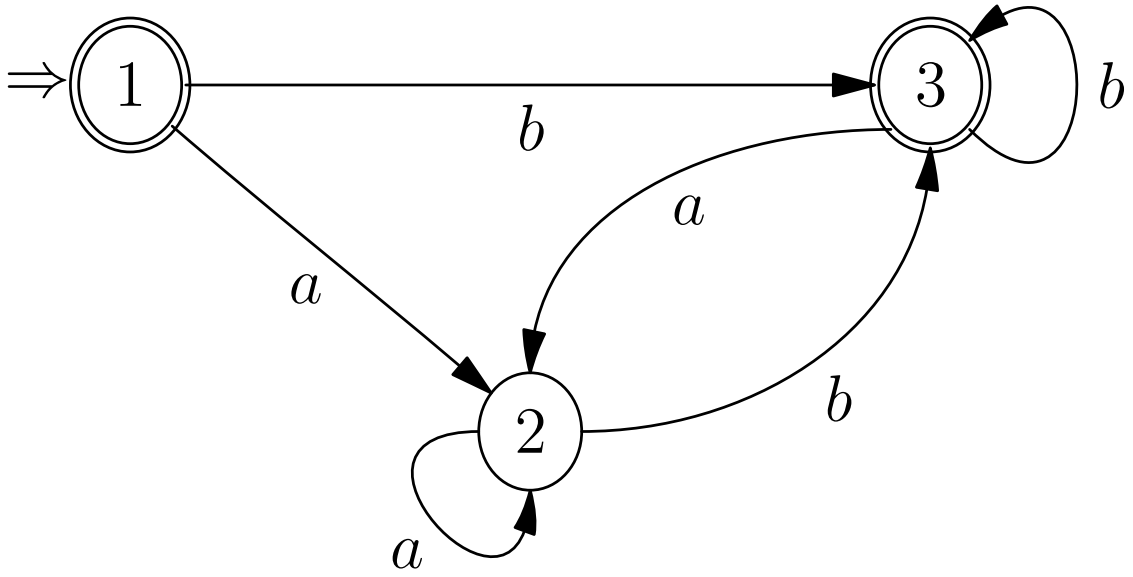
Méthode de l'arrivée

C'est l'inverse de la méthode précédente :

On note A_k l'ensemble des mots qui font arriver à l'état k en partant de l'état initial

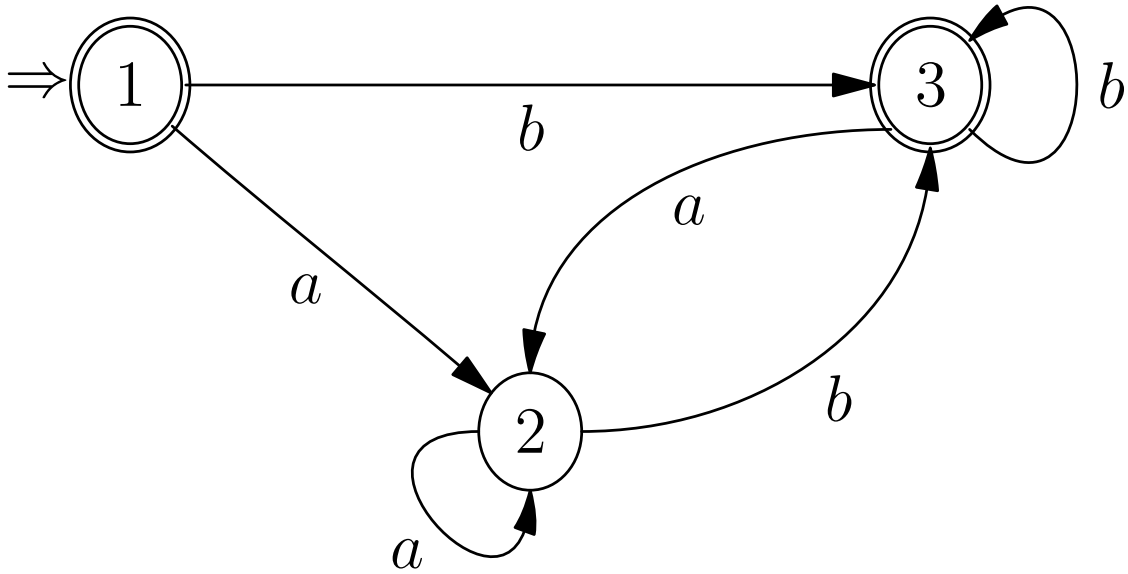
Les A_k sont liés par des équations (une par état)

Méthode de l'arrivée



On a :

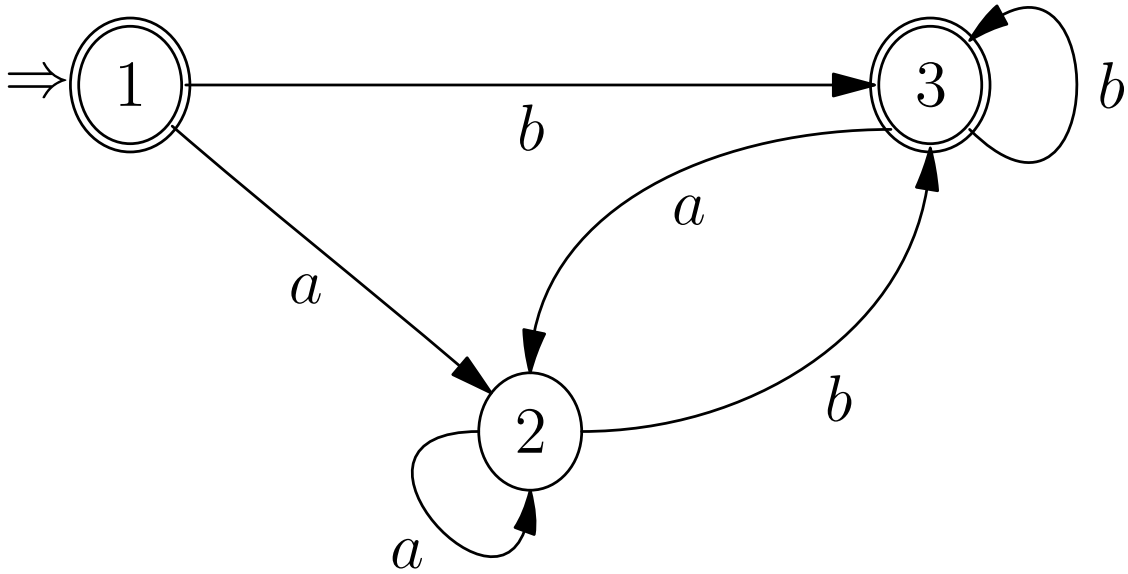
Méthode de l'arrivée



On a :

- $A_1 = \varepsilon$

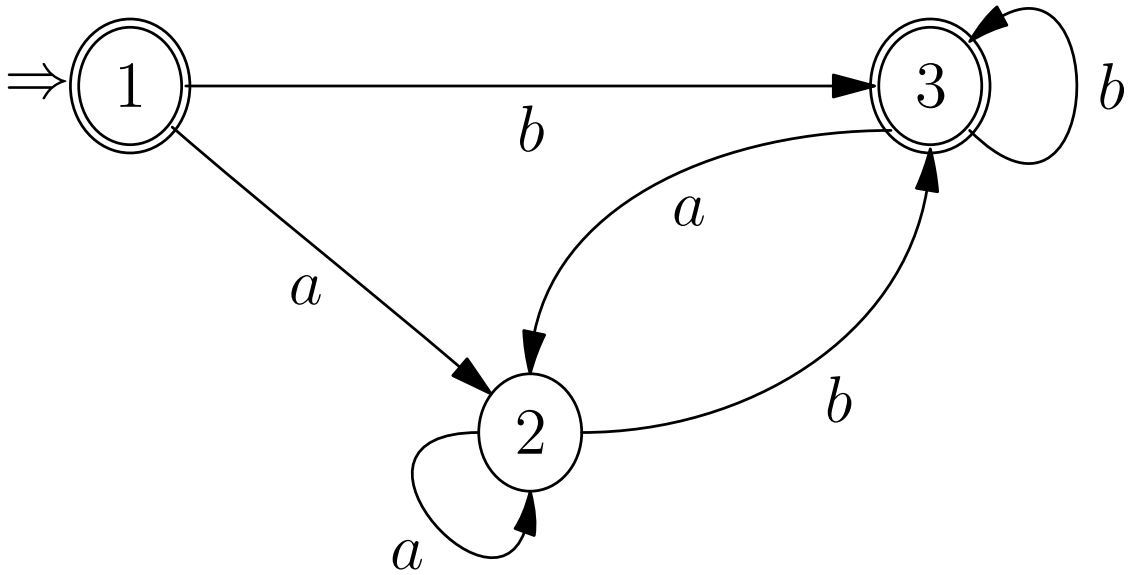
Méthode de l'arrivée



On a :

- $A_1 = \varepsilon$
- $A_2 = A_1 a + A_2 a + A_3 a$

Méthode de l'arrivée



On a :

- $A_1 = \varepsilon$
- $A_2 = A_1 a + A_2 a + A_3 a$
- $A_3 = A_1 b + A_2 b + A_3 b$

Lemme D'Arden

La résolution des ces systèmes repose sur le lemme d'Arden :

Lemme D'Arden

La résolution des ces systèmes repose sur le lemme d'Arden :

Théorème (Lemme D'Arden)

Soit U et V deux langages et les équations

$$X = UX + V \quad (1) \quad \textit{et} \quad X = XU + V \quad (2)$$

d'inconnue X

Lemme D'Arden

La résolution des ces systèmes repose sur le lemme d'Arden :

Théorème (Lemme D'Arden)

Soit U et V deux langages et les équations

$$X = UX + V \quad (1) \quad \textit{et} \quad X = XU + V \quad (2)$$

d'inconnue X

1. *le plus petit langage solution de* (1) *est* $X = U^*V$

Lemme D'Arden

La résolution des ces systèmes repose sur le lemme d'Arden :

Théorème (Lemme D'Arden)

Soit U et V deux langages et les équations

$$X = UX + V \quad (1) \quad \textit{et} \quad X = XU + V \quad (2)$$

d'inconnue X

1. *le plus petit langage solution de* (1) *est* $X = U^*V$
2. *le plus petit langage solution de* (2) *est* $X = VU^*$

Lemme D'Arden

La résolution des ces systèmes repose sur le lemme d'Arden :

Théorème (Lemme D'Arden)

Soit U et V deux langages et les équations

$$X = UX + V \quad (1) \quad et \quad X = XU + V \quad (2)$$

d'inconnue X

1. *le plus petit langage solution de (1) est $X = U^*V$*
2. *le plus petit langage solution de (2) est $X = VU^*$*
3. *De plus, si $\varepsilon \notin U$, alors ces solutions sont uniques*

Méthode du départ

Résolvons le système pour la méthode du départ :

$$\begin{cases} D_1 = \varepsilon + aD_2 + bD_3 \\ D_2 = aD_2 + bD_3 \\ D_3 = \varepsilon + aD_2 + bD_3 \end{cases}$$

Méthode du départ

Résolvons le système pour la méthode du départ :

$$\begin{cases} D_1 = \varepsilon + aD_2 + bD_3 \\ D_2 = aD_2 + bD_3 \\ D_3 = \varepsilon + aD_2 + bD_3 \end{cases}$$

Le 3^{e} équation peut s'écrire $D_3 = bD_3 + (aD_2 + \varepsilon)$

Méthode du départ

Résolvons le système pour la méthode du départ :

$$\begin{cases} D_1 = \varepsilon + aD_2 + bD_3 \\ D_2 = aD_2 + bD_3 \\ D_3 = \varepsilon + aD_2 + bD_3 \end{cases}$$

Le 3^e équation peut s'écrire $D_3 = bD_3 + (aD_2 + \varepsilon)$
Si on considère que D_3 est l'unique inconnue de cette équation, le lemme d'Arden donne :

$$D_3 = b^*(aD_2 + \varepsilon) = b^*aD_2 + b^*$$

Méthode du départ

En reportant dans les deux premières équations, on obtient :

$$D_1 = aD_2 + bb^*aD_2 + bb^* = (\varepsilon bb^*)aD_2 + b^+ = b^*aD_2 + b^+$$
$$D_2 = \varepsilon + aD_2 + bb^*aD_2 + bb^* = (\varepsilon + bb^*)aD_2 + b^+ + \varepsilon = b^*aD_2 + b^*$$

Méthode du départ

En reportant dans les deux premières équations, on obtient :

$$D_1 = aD_2 + bb^*aD_2 + bb^* = (\varepsilon bb^*)aD_2 + b^+ = b^*aD_2 + b^+$$
$$D_2 = \varepsilon + aD_2 + bb^*aD_2 + bb^* = (\varepsilon + bb^*)aD_2 + b^+ + \varepsilon = b^*aD_2 + b^*$$

On recommence avec la deuxième équation et on obtient :

$$D_2 = (b^*a)^*b^+$$

Méthode du départ

Finalement,

$$L = D_1 = (b^*a)(b^*a)^*b^* + b^* = (b^*a)^*b^+ + \varepsilon$$

Méthode du départ

Finalement,

$$L = D_1 = (b^*a)(b^*a)^*b^* + b^* = (b^*a)^*b^+ + \varepsilon$$

Remarque

1. Avec la méthode de l'arrivée, on trouve

$$L = (a^*b)^*$$

Méthode du départ

Finalement,

$$L = D_1 = (b^*a)(b^*a)^*b^* + b^* = (b^*a)^*b^+ + \varepsilon$$

Remarque

1. Avec la méthode de l'arrivée, on trouve

$$L = (a^*b)^*$$

2. un même langage peut être décrit par deux formules différentes qui ne se ressemblent pas

Sommaire

Expression régulière

Les méthodes pour déterminer le langage d'un automate fini aboutissent à une formule combinant les lettres de l'alphabet Σ en utilisant :

Expression régulière

Les méthodes pour déterminer le langage d'un automate fini aboutissent à une formule combinant les lettres de l'alphabet Σ en utilisant :

- l'addition ;

Expression régulière

Les méthodes pour déterminer le langage d'un automate fini aboutissent à une formule combinant les lettres de l'alphabet Σ en utilisant :

- l'addition ;
- la concaténation ;

Expression régulière

Les méthodes pour déterminer le langage d'un automate fini aboutissent à une formule combinant les lettres de l'alphabet Σ en utilisant :

- l'addition ;
- la concaténation ;
- l'étoile

Expression régulière

Les méthodes pour déterminer le langage d'un automate fini aboutissent à une formule combinant les lettres de l'alphabet Σ en utilisant :

- l'addition ;
- la concaténation ;
- l'étoile

Une telle formule s'appelle une *expression régulière*.

Expression régulière

Les méthodes pour déterminer le langage d'un automate fini aboutissent à une formule combinant les lettres de l'alphabet Σ en utilisant :

- l'addition ;
- la concaténation ;
- l'étoile

Une telle formule s'appelle une *expression régulière*.

Plus précisément, une expression régulière est un mot construit avec l'alphabet

$$\Theta = \{+, *, (,), \varepsilon, \emptyset\} \cup \Sigma$$

Expression régulière

Règles de bases

Expression régulière

Règles de bases

R1 : ε et $\emptyset$ sont des expressions régulières

Expression régulière

Règles de bases

R1 : ε et $\emptyset$ sont des expressions régulières

R2 : si a est une lettre, a est une expression régulière

Expression régulière

Règles de bases

R1 : ε et $\emptyset$ sont des expressions régulières

R2 : si a est une lettre, a est une expression régulière

Règle de combinaison

R3 : Si $\mathcal{E}$ est une expression régulière, alors $(\mathcal{E})$ est une expression régulière

Expression régulière

Règles de bases

R1 : ε et $\emptyset$ sont des expressions régulières

R2 : si a est une lettre, a est une expression régulière

Règle de combinaison

R3 : Si $\mathcal{E}$ est une expression régulière, alors $(\mathcal{E})$ est une expression régulière

R4 : Si $\mathcal{E}$ est une expression régulière, alors $\mathcal{E}^*$ est une expression régulière

Expression régulière

Règles de bases

R1 : ε et $\emptyset$ sont des expressions régulières

R2 : si a est une lettre, a est une expression régulière

Règle de combinaison

R3 : Si $\mathcal{E}$ est une expression régulière, alors $(\mathcal{E})$ est une expression régulière

R4 : Si $\mathcal{E}$ est une expression régulière, alors $\mathcal{E}^*$ est une expression régulière

R5 Si $\mathcal{E}$ et $\mathcal{F}$ sont des expressions régulières, alors $(\mathcal{E}) + (\mathcal{F})$ est une expression régulière

Expression régulière

Règles de bases

R1 : ε et $\emptyset$ sont des expressions régulières

R2 : si a est une lettre, a est une expression régulière

Règle de combinaison

R3 : Si $\mathcal{E}$ est une expression régulière, alors $(\mathcal{E})$ est une expression régulière

R4 : Si $\mathcal{E}$ est une expression régulière, alors $\mathcal{E}^*$ est une expression régulière

R5 Si $\mathcal{E}$ et $\mathcal{F}$ sont des expressions régulières, alors $(\mathcal{E}) + (\mathcal{F})$ est une expression régulière

R6 : Si $\mathcal{E}$ et $\mathcal{F}$ sont des expressions régulières, alors $(\mathcal{E})(\mathcal{F})$ est une expression régulière

Langage régulier

Définition

Un langage qui peut-être décrit au moyen d'une expression régulière est un *langage régulier*

Langage régulier

Définition

Un langage qui peut-être décrit au moyen d'une expression régulière est un *langage régulier*

Théorème

Tout langage accepté par un automate fini est régulier

Langage Régulier

2 questions :

Q1 : Tous les langages sont-ils réguliers ?

Q2 : Tous les langages réguliers sont-ils acceptés par un automate ?

Langage Régulier

2 questions :

Q1 : Tous les langages sont-ils réguliers ?

- Non, les langages sont classés en 4 catégories (type0, type1, ...)

Q2 : Tous les langages réguliers sont-ils acceptés par un automate ?

Langage Régulier

2 questions :

Q1 : Tous les langages sont-ils réguliers ?

- Non, les langages sont classés en 4 catégories (type0, type1, ...)

Q2 : Tous les langages réguliers sont-ils acceptés par un automate ?

- Oui, tout langage régulier est celui d'un automate fini

Sommaire

Problématique

- On sait déjà déterminer l'expression rationnelle d'un automate fini ;

Problématique

- On sait déjà déterminer l'expression rationnelle d'un automate fini ;
- Réciproquement, à partir de l'expression rationnelle d'un langage régulier, comment construire un automate fini (déterministe complet) qui accepte ce langage ?

Problématique

- On sait déjà déterminer l'expression rationnelle d'un automate fini ;
- Réciproquement, à partir de l'expression rationnelle d'un langage régulier, comment construire un automate fini (déterministe complet) qui accepte ce langage ?
- Comment arriver à l'automate le plus simple possible ?

Problématique

- On sait déjà déterminer l'expression rationnelle d'un automate fini ;
- Réciproquement, à partir de l'expression rationnelle d'un langage régulier, comment construire un automate fini (déterministe complet) qui accepte ce langage ?
- Comment arriver à l'automate le plus simple possible ?

On utilise les *résiduels* du langage

Résiduels d'un langage

Définition

Soient Σ un alphabet, L un langage sur Σ et $\sigma \in \Sigma^*$. On appelle *résiduel de L par rapport à σ* le langage formé de tous les mots τ tels que :

$$\sigma \cdot \tau \in L$$

On le note $\sigma^{-1}L$.

Résiduels d'un langage

Définition

Soient Σ un alphabet, L un langage sur Σ et $\sigma \in \Sigma^*$. On appelle *résiduel de L par rapport à σ* le langage formé de tous les mots τ tels que :

$$\sigma \cdot \tau \in L$$

On le note $\sigma^{-1}L$.

Remarque

D'un point de vue concret, $\sigma^{-1}L$ est le langage obtenu en prenant les mots de L commençant par σ et en effaçant ce σ au début des mots.

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$

Soit $s \in \Sigma^*$.

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$
Soit $s \in \Sigma^*$.

1. Si b est une lettre de σ,

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$
Soit $s \in \Sigma^*$.

1. Si b est une lettre de σ,
   - il n'y a pas de mot de L commençant par σ ;

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$
Soit $s \in \Sigma^*$.

1. Si b est une lettre de σ,
   - il n'y a pas de mot de L commençant par σ ;
   - $\sigma^{-1}L = \emptyset$

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$
Soit $s \in \Sigma^*$.

1. Si b est une lettre de σ,
   - il n'y a pas de mot de L commençant par σ ;
   - $\sigma^{-1}L = \emptyset$
2. Si b n'est pas une lettre de σ,

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$
Soit $s \in \Sigma^*$.

1. Si b est une lettre de σ,
   - il n'y a pas de mot de L commençant par σ ;
   - $\sigma^{-1}L = \emptyset$
2. Si b n'est pas une lettre de σ,
   - $\sigma = a^n$

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$
Soit $s \in \Sigma^*$.

1. Si b est une lettre de σ,
   - il n'y a pas de mot de L commençant par σ ;
   - $\sigma^{-1}L = \emptyset$
2. Si b n'est pas une lettre de σ,
   - $\sigma = a^n$
   - les mots de L commençant par σ s'écrivent a^{n+m} où $m \geqslant 0$

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$
Soit $s \in \Sigma^*$.

1. Si b est une lettre de σ,
   - il n'y a pas de mot de L commençant par σ ;
   - $\sigma^{-1}L = \emptyset$
2. Si b n'est pas une lettre de σ,
   - $\sigma = a^n$
   - les mots de L commençant par σ s'écrivent a^{n+m} où $m \geqslant 0$
   - en enlevant a^n on obtient a^m avec $m \geqslant 0$

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$
Soit $s \in \Sigma^*$.

1. Si b est une lettre de σ,
   - il n'y a pas de mot de L commençant par σ ;
   - $\sigma^{-1}L = \emptyset$
2. Si b n'est pas une lettre de σ,
   - $\sigma = a^n$
   - les mots de L commençant par σ s'écrivent a^{n+m} où $m \geqslant 0$
   - en enlevant a^n on obtient a^m avec $m \geqslant 0$
   - et $\sigma^{-1}L = L$

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$
Soit $s \in \Sigma^*$.

1. Si b est une lettre de σ,
   - il n'y a pas de mot de L commençant par σ ;
   - $\sigma^{-1}L = \emptyset$
2. Si b n'est pas une lettre de σ,
   - $\sigma = a^n$
   - les mots de L commençant par σ s'écrivent a^{n+m} où $m \geqslant 0$
   - en enlevant a^n on obtient a^m avec $m \geqslant 0$
   - et $\sigma^{-1}L = L$

L admet 2 résiduels qui sont L et $\emptyset$

Résiduels d'un langage

Méthode pratique de construction

Résiduels d'un langage

Méthode pratique de construction

1. les états sont les différents résiduels de L ;

Résiduels d'un langage

Méthode pratique de construction

1. les états sont les différents résiduels de L ;
2. l'état initial est L ;

Résiduels d'un langage

Méthode pratique de construction

1. les états sont les différents résiduels de L ;
2. l'état initial est L ;
3. les états acceptants sont les résiduels qui contiennent ε ;

Résiduels d'un langage

Méthode pratique de construction

1. les états sont les différents résiduels de L ;
2. l'état initial est L ;
3. les états acceptants sont les résiduels qui contiennent ε ;
4. la flèche a partant du résiduel R arrive au résiduel $a^{-1}R$.

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$

- L admet deux résiduels L et $\emptyset$;

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$

- L admet deux résiduels L et $\emptyset$;
- l'automate minimal de L admet deux états ;

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$

- L admet deux résiduels L et $\emptyset$;
- l'automate minimal de L admet deux états ;
- L est l'état initial et l'unique état acceptant ;

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$

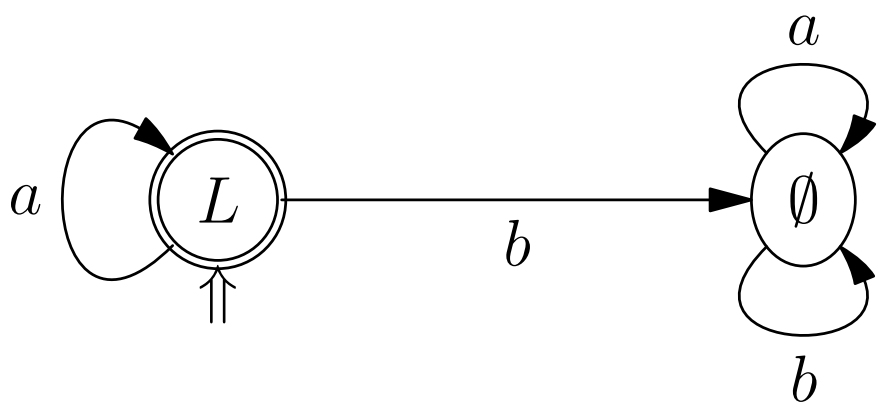
- L admet deux résiduels L et $\emptyset$;
- l'automate minimal de L admet deux états ;
- L est l'état initial et l'unique état acceptant ;
- $a^{-1}L = L$, $b^{-1}L = \emptyset$, $a^{-1}\emptyset = \emptyset$ et $b^{-1}\emptyset = \emptyset$

Résiduels d'un langage

Exemple

$\Sigma = \{a, b\}$ et $L = a^* = \{\varepsilon, a, a^2, a^3, \ldots\}$

- L admet deux résiduels L et $\emptyset$;
- l'automate minimal de L admet deux états ;
- L est l'état initial et l'unique état acceptant ;
- $a^{-1}L = L$, $b^{-1}L = \emptyset$, $a^{-1}\emptyset = \emptyset$ et $b^{-1}\emptyset = \emptyset$



Résiduels d'un langage

Théorème

1. *L'automate $\mathcal{M}$ construit par cette méthode admet L pour langage ;*

Résiduels d'un langage

Théorème

1. *L'automate $\mathcal{M}$ construit par cette méthode admet L pour langage ;*
2. *Si R est un résiduel de L, alors le langage de départ de l'état R est R ;*

Résiduels d'un langage

Théorème

1. *L'automate $\mathcal{M}$ construit par cette méthode admet L pour langage ;*
2. *Si R est un résiduel de L, alors le langage de départ de l'état R est R ;*
3. *Parmi tous les automates qui admettent L pour langage, c'est $\mathcal{M}$ qui possède le moins d'états.*

Résiduels d'un langage

Théorème

1. *L'automate $\mathcal{M}$ construit par cette méthode admet L pour langage ;*
2. *Si R est un résiduel de L, alors le langage de départ de l'état R est R ;*
3. *Parmi tous les automates qui admettent L pour langage, c'est $\mathcal{M}$ qui possède le moins d'états.*

L'automate $\mathcal{M}$ s'appelle l'automate minimal de L

Résiduels d'un langage

Théorème

1. *L'automate $\mathcal{M}$ construit par cette méthode admet L pour langage ;*
2. *Si R est un résiduel de L, alors le langage de départ de l'état R est R ;*
3. *Parmi tous les automates qui admettent L pour langage, c'est $\mathcal{M}$ qui possède le moins d'états.*

L'automate $\mathcal{M}$ s'appelle l'automate minimal de L

Remarque

On utilise aussi cette méthode pour simplifier les automates

Automates finis déterministes complets

Pour l'instant nous avons rencontré des automates finis ayant :

- un unique seul état initial ;

Automates finis déterministes complets

Pour l'instant nous avons rencontré des automates finis ayant :

- un unique seul état initial ;
- pour chaque état, exactement une transition par label

Automates finis déterministes complets

Pour l'instant nous avons rencontré des automates finis ayant :

- un unique seul état initial ;
- pour chaque état, exactement une transition par label

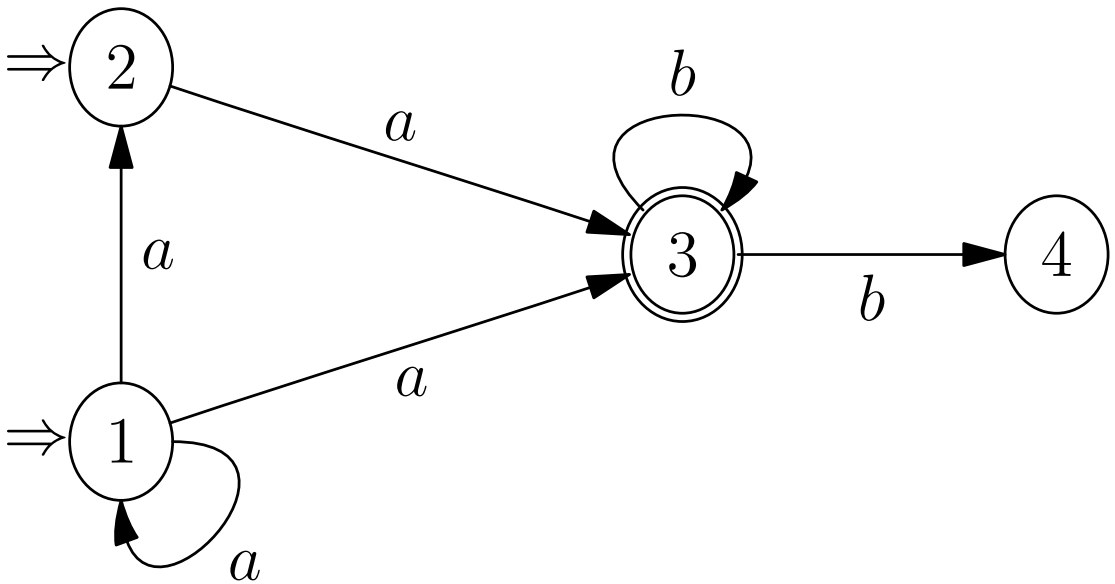
Ce sont les automates finis déterministes complets (AFD)

Automates finis non déterministes

Dans la pratique, il peut-être utile de concevoir des automates *moins rigide*

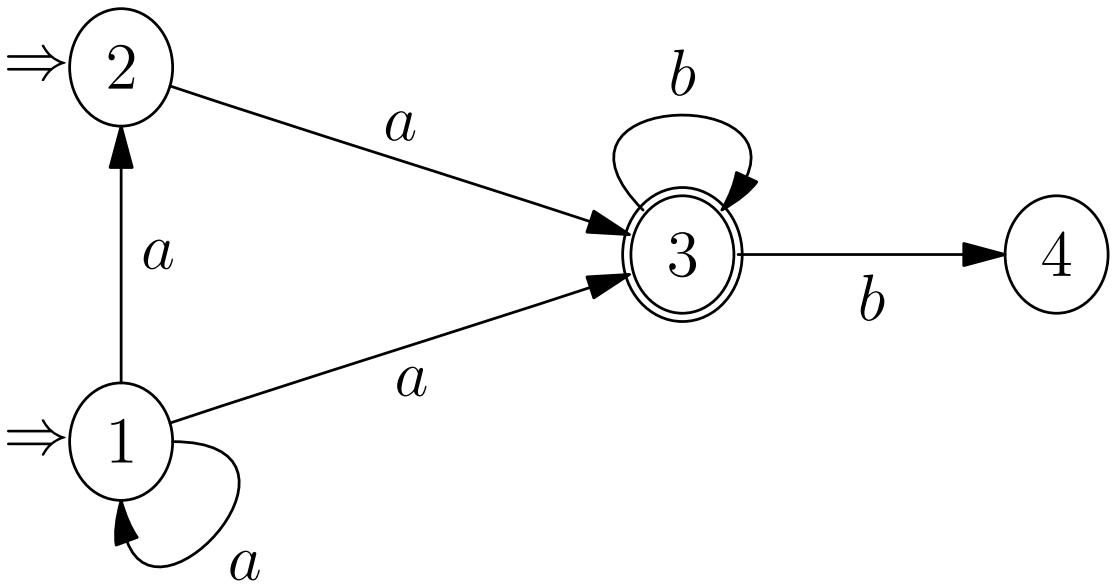
Automates finis non déterministes

Dans la pratique, il peut-être utile de concevoir des automates *moins rigide*



Automates finis non déterministes

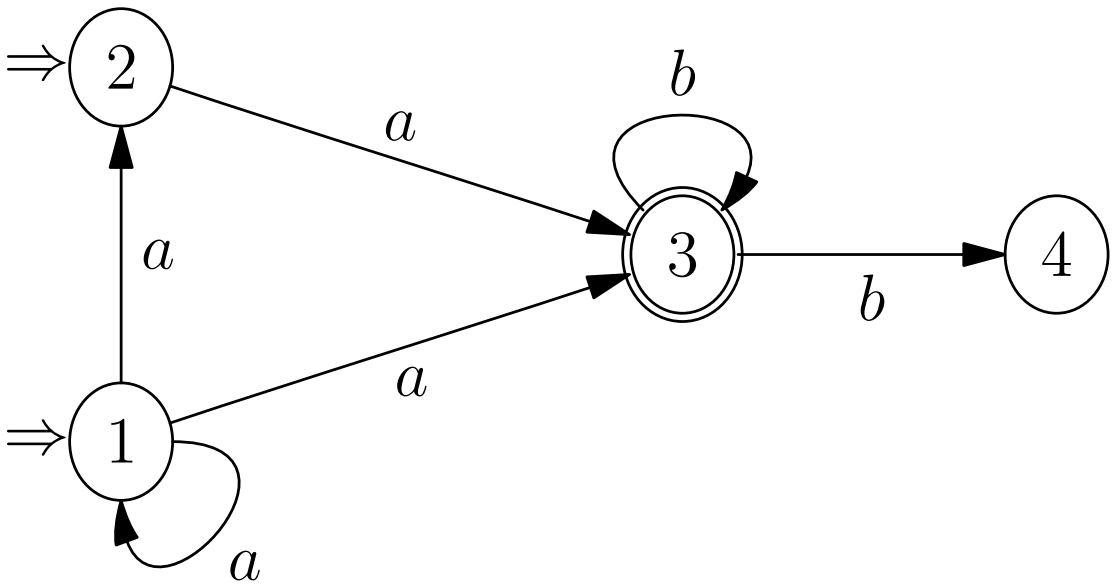
Dans la pratique, il peut-être utile de concevoir des automates *moins rigide*



➢ ni déterministe, ni complet

Automates finis non déterministes

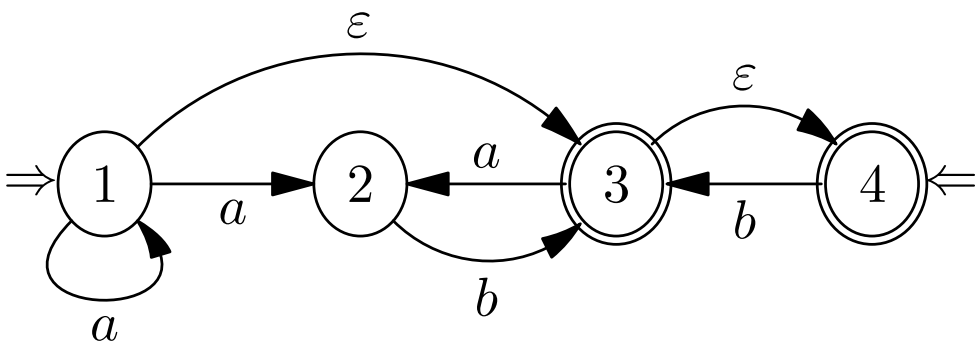
Dans la pratique, il peut-être utile de concevoir des automates *moins rigide*



- ni déterministe, ni complet
- automate non déterministe (AFN)

Transitions spontannées

On peut aussi utiliser les *transitions spontanées* :



Automates finis non déterministes

Théorème

1. *les langages des AFN sont réguliers ;*

Automates finis non déterministes

Théorème

1. *les langages des AFN sont réguliers ;*
2. *les langages acceptés par les AFN et les AFD sont les mêmes ;*

Automates finis non déterministes

Théorème

1. *les langages des AFN sont réguliers ;*
2. *les langages acceptés par les AFN et les AFD sont les mêmes ;*

Remarque (Détermination d'un AFN)

On peut ainsi, à partir d'un AFN, trouver un AFD qui reconnait le même langage

Theorème de Kleene

Théorème (Kleene)

les langages acceptés par les automates finis sont tous les langages réguliers