Automates

Lundi 30 mars

Option Informatique

Ecole Alsacienne

AVANT DE COMMENCER…

- Trolls et châteaux :
  - Prochain boîte de gâteaux : 4 mai
  - Note pour le troisième trimestre : classement mixte

- Prochaines séances
  - Lundi 6 avril : lundi de Pâques
  - Lundi 13 avril : programmation orientée objet

- Projet de programmation
  - Seul ou en binôme
  - Présentation à la fin de l'année
    - Présentation du sujet
    - Difficultés rencontrées
    - Choix effectués et solutions trouvées
    - Démonstration du programme
  - Pour le 13 avril : proposition de groupes et de sujets

- Livret scolaire / APB

- Reprise à partir du slide 57

Plan

1. Motivations
2. Alphabets et langages
3. Expressions rationnelles
4. Automates finis déterministes
5. Automates finis non déterministes
6. ε-Transitions
7. Minimisation
8. Automates finis et expressions rationnelles

MOTIVATIONS

QUELQUES LIGNES DE CODE

- Que fait le code ci-dessous ?

```
x = 1
for i in range(10):
    x = i + x / 2 + 1
print(x)
```

- Comment en êtes-vous arrivés à ces conclusions ?

```
x = 1
for i in range(10):
    x = i + x * 2 + 1
print(x)
```

Quelques lignes de code

```
x = 1
for i in range(10):
    x = i + x * 2 + 1
print(x)
```

- Ces quelques lignes de codes contiennent beaucoup d'informations :
  - Noms de variables
  - Noms de fonctions
  - Valeurs numériques
  - Déclaration de boucle et corps de boucle
  - Symboles divers (parenthèses, opérateurs, signe d'égalité)
  - Priorité entre opérateurs (le * intervient avant le +)

Quelques lignes de code

```
x = 1
for i in range(10):
    x = i + x * 2 + 1
print(x)
```

- Avec l'habitude, vous déduisez toutes ces informations sans même y penser.

- **Question :** Comment un ordinateur est-il capable d'effectuer cette même analyse ?

Syntax error

- Que fait le code ci-dessous ?

```
x = 1
for i in range(10)
    x = i    x / 2 + 1
        print(i)
print(x))
```

- **Réponse** : Il plante !
  - Il manque un point-virgule après `range(10)`
  - Il manque un symbole entre `i` et `x`
  - L'instruction `print(i)` est mal indentée
  - Il y a une parenthèse en trop sur la dernière ligne

- **Question** : Comment un ordinateur est-il capable d'identifier une erreur de syntaxe ?

Analyse syntaxique

- L'analyse syntaxique consiste à étudier une suite de symboles pour essayer d'en déduire la structure et le sens

- Cette analyse permet en outre de détecter certaines erreurs (ex : symbole oublié)

- **Remarque :** Les erreurs de syntaxe ne sont pas les seules possibles ; même si la syntaxe est valide, un programme ne fait pas forcément ce que l'on croît.

```
x = 1
for i in range(10):
    x = i + x / 2 + 1
    print(x)
```

Liens avec les langages naturels

- La notion d'analyse syntaxique existe également avec les langages naturels.

- Là encore, il s'agit de donner du sens à une suite de symboles.
  - Exemple : *Il y a deux réponses à cette question.*
  - L'esprit tente d'identifier un verbe, un sujet, des compléments, etc.

- Il est également possible de faire des erreurs :
  - Fautes d'orthographe : *Il y a deux réponse à cet question.*
  - Construction incorrecte : *Il y a réponses deux à cette question.*
  - Etc.

Liens avec les langages naturels

- Cette analyse est très complexe pour les langages naturels
  - Un pan entier de la recherche en informatique y est consacré
  - Exemples de thématique :
    - extraction du sens
    - traduction
    - analyse des sentiments

- Mais les difficultés sont nombreuses :
  - Polysémie : *Les poules du couvent couvent.*
  - Négation : *Nul n'est censé ignorer la loi.*
  - Ironie : *Quel splendide temps normand !*
  - Expressions figurées : *Il pleut des cordes.*
  - Fautes d'orthographe : *Je pnese donc je ssuis.*
  - Tournures inattendues : *Toujours par deux ils vont.*
  - Etc.

Recherche de motif

- **Question :** Comment chercher la présence des termes suivants dans un texte ?
  - Mots qui commencent par la lettre A ?
  - Mots de 3 lettres qui comment par la lettre A ?
  - Mots qui commencent par la lettre A et qui se terminent par ENT ?
  - Mots composés d'exactement 5 chiffres ?
  - Mots construits selon le modèle suivant *[nom]@[domaine].[extension]*

Recherche de motif

- Exemples d'utilisation :
  - Rechercher des séquences de nucléotides dans un brin d'ADN (par exemple pour détecter des maladies)
  - Retrouver un code postal dans un gros volume de texte
  - Vérifier qu'une chaîne de caractères entrée par un utilisateur correspond bien à un email
  - Remplacer des centaines d'occurrences d'un seul coup

    Exemple : `EA_`*`prenom_nom`*`.doc` devient `EA_nom_prenom.doc`
  - Etre capable d'exploiter toute la puissance des expressions régulières

Une nouvelle séquence

- Tous ces sujets sont en fait liés à une notion très utile en informatique : les langages rationnels.

- Vous avez peut-être rencontré certains termes qui se rapprochent de cette idée :
  - Langages réguliers
  - Expressions rationnelles
  - Expressions régulières ("regex")
  - Automates finis (déterministes ou non)

- C'est justement le thème de cette nouvelle séquence !

Alphabets et langages

Alphabets

- Il existe de nombreux alphabets :
  - A B C D E
  - α β γ δ ε
  - ا ب ت ٹ ج
  - 你说汉语吗
  - 
  - etc.

Définitions

- Un alphabet est un ensemble fini de symboles
  - Par convention, cet alphabet est souvent noté Σ
- Un mot sur l'alphabet Σ est une suite finie (eventuellement vide) d'éléments de Σ.
- Par convention, le mot vide est noté ε.
- La longueur d'un mot u est notée $|u|$.
  - On a notamment $|\varepsilon| = 0$

Nombre d'occurrences

- Le nombre d'occurrence du symbole a dans le mot u est noté $|u|_a$

- Propriété directe :

$$|u| = \sum_{a \in \Sigma} |u|_a$$

- On note a^n le mot composé de n occurrences du symbole a
  - On a notamment $a^0 = \varepsilon$

- La concaténation des mots u et v est notée uv.

LANGAGES

- L'ensemble de tous les mots formés à partir de l'alphabet Σ est noté $\Sigma^\star$.

- L'ensemble de tous les mots non vides formés à partir de l'alphabet Σ est noté Σ^+.

- Un langage sur l'alphabet Σ est un sous-ensemble de $\Sigma^\star$
  - Par convention, un langage est souvent noté L, ou L_1, etc.

Premiers exemples de langages

- Considérons l'alphabet $\Sigma = \{\mathrm{a}, \mathrm{b}\}$.

- Les exemples ci-dessous sont tous des langages sur cet alphabet Σ :
  - $\Sigma^\star$: Tous les mots composés des lettres a ou b (y compris le mot vide)
  - $L_1 = \{aa, ab, ba, bb\}$: Tous les mots de longueur 2
  - $L_2 = \{ab, aab, abb, aaab, \ldots\}$: Tous les mots qui commencent par a et qui se terminent par b
  - $L_3 = \{a^n b^n \mid n \geq 0\}$: Tous les mots composés d'une succession de a puis du même nombre de b
  - $L_4 = \{(aa)^n \mid n \geq 0\}$: Tous les mots composés d'un nombre pair de a (ce qui inclut le mot vide)

Des possibilités infinies

- Un alphabet est un ensemble fini de symboles

- Pourtant,
  - Il existe une infinité de mots pour un alphabet donné
  - Il existe une infinité de langages sur un alphabet donné

- Un langage fini est un langage qui contient un nombre fini de mots
  - Exemple : $L_1 = \{aa, ab, ba, bb\}$: Tous les mots de longueur 2
  - En général, les langages étudiés contiennent un nombre infini de mots

Opérations ensemblistes

- L'union de deux langages L_1 et L_2 est le langage constitué des mots appartenant à L_1 ou à L_2
  - C'est un "OU" non exclusif
  - Cette union est souvent notée $L_1 \cup L_2$

- L'intersection de deux langages L_1 et L_2 est le langage constitué des mots appartenant à la fois à L_1 et à L_2
  - Cette intersection est souvent notée $L_1 \cap L_2$

- Le complémentaire (dans Σ^*) d'un langage L est le langage constitué de l'ensemble des mots n'appartenant pas à L.
  - Ce complémentaire est souvent noté $\overline{L}$

Opérations ensemblistes

- Pour tout langage L sur $\Sigma^\star$, on a
  - $\bar{L} \cup L = \Sigma^\star$
  - $\bar{L} \cap L = \emptyset$ (ensemble vide)

- La concaténation de deux langages L_1 et L_2 est le langage constitué des mots formés d'un mot de L_1 suivi d'un mot de L_2
  - Cette concaténation est souvent notée$L_1 L_2 L_2$
  - Formellement, $L_1 L_2 = \{uv \mid u \in L_1, v \in L_2\}$

Opérations ensemblistes

- Si L est un langage, le langage L^n est le langage constitué des mots formés de n mots de L
  - Formellement, $L^n = \{u = u_1 u_2 \cdots u_n \mid u_1 \in L, u_2 \in L, \cdots, u_n \in L\}$
  - On a notamment $L^0 = \{\varepsilon\}$
  - A ne pas confondre avec le langage $\{u = v^n \mid v \in L\}$ (qui ne contient que les puissances n$^{\text{ièmes}}$ des mots de L)

- L'étoile (aussi appelée *fermeture de Kleene*) d'un langage L est notée $L^\star$, et est définie par

$$L^\star = \bigcup_{i \geq 0} L^i$$

  - $L^\star$ contient donc tous les mots qu'il est possible de construire en concaténant un nombre fini de mots de L.

EXPRESSIONS RATIONNELLES

DÉFINITION RÉCURSIVE

- Les expressions rationnelles sur un alphabet Σ sont définies de la façon suivante :
  - L'ensemble vide $\emptyset$ est une expression rationnelle
  - Le mot vide ε est une expression rationnelle
  - Pour tout symbole $a \in \Sigma$, a est une expression rationnelle
  - Si e_1 et e_2 sont des expressions rationnelles,
    - L'union $e_1 + e_2$ est une expression rationnelle
    - La concaténation $e_1 e_1$ est une expression rationnelle
    - L'étoile ${e_1}^\star$ est une expression rationnelle

- Remarques :
  - C'est une définition récursive
  - A chaque expression régulière peut être associé un langage

Premier exemple

- Alphabet : $\Sigma = \{\mathrm{a}, \mathrm{e}, \mathrm{f}, \mathrm{i}, \mathrm{r}\}$.

- Les exemples ci-dessous sont autant d'expressions rationnelles :
  - a, e, f, i, r : langages constitués d'un seul symbole
  - (fa) : langage constitué du mot fa, concaténation de f et a
  - $(\big(\big((fa)i\big)r\big)\,e)$: langage constitué du mot $faire$
  - (re) : langage constitué du mot re
  - $(re)^\star$: langage constitué des concaténations d'un nombre fini de mots de (re)
    - C'est-à-dire ε, re, $rere$, $rerere$, etc.
  - $(re)^\star(\big(\big((fa)i\big)r\big)\,e)$: langage constitué d'un certain nombre d'occurrence du préfixe re, suivi du mot $faire$
    - C'est-à-dire $faire$, $refaire$, $rerefaire$, $rererefaire$, etc.

Convention d'écriture

- Dans un souci de lisibilité, on utilise souvent les conventions suivantes :
  - L'étoile ($\star$) est prioritaire sur tous les autres opérateurs
  - La concaténation ($\cdot$) est prioritaire sur l'union ($+$)

- Cela permet de supprimer de nombreuses parenthèses pour alléger la lecture
  - $(re)^\star(\Big(\big((fa)i\big)r\Big)e)$ devient $(re)^\star faire$
  - $\Big(\big((ba)(a)^\star\big)(b)^\star\Big)b$ devient $baa^\star b^\star b$
  - etc.

Expressions et langages

- **Question** : Quels sont les langages représentés par les expressions rationnelles ci-dessous :
  - $a(a+b)^\star b$
  - $a(a^\star b^\star)^\star b$
  - $a(a^\star b)^\star a^\star b$

- Deux expressions peuvent représenter le même langage !

- Il est parfois possible de simplifier une expression rationnelle

  Si e est une expression rationnelle,
  - $e + e = e$
  - $(e^\star)^\star = e^\star$
  - etc.

Exercices

- **Exercice 1** : A quoi correspondent les expressions rationnelles suivantes sur l'alphabet $\Sigma = \{a, b, c\}$.
  1. $a(a + b + c)^\star$
  2. $(a + bb + c)^\star$
  3. $(a + b + c)(a + b + c)$

- **Exercice 2** : Peut-on trouver une expression rationnelle pour décrire les langages suivants ?
  1. Mots qui ne commencent pas par a
  2. Mots qui contiennent un nombre impair de c
  3. Mots qui contiennent exactement 3 a
  4. Mots qui contiennent autant de b que de c

Expressions régulières

- On parle également souvent d'expressions régulières.

- Ces expressions ressemblent aux expressions rationnelles, avec plusieurs symboles supplémentaires :
  - `^` début de ligne
  - `$` fin de ligne
  - `\d` n'importe quel chiffre
  - `\D` n'importe quel caractère autre qu'un chiffre
  - `\w` n'importe quel caractère alphabétique (lettre, chiffre, underscore)
  - `c{n}` le caractère `c` exactement `n` fois
  - etc.

Expressions régulières

- Ces expressions régulières incluent également la notion de "regroupement".

- On peut ainsi
  - "Capturer" une partie d'un motif
  - Vérifier qu'un même sous-motif apparaît plusieurs fois (répétition)
  - Vérifier qu'un sous-motif n'apparaît pas
  - etc.

- Les expressions régulières sont un outil extrêmement puissant
  - Elles permettent de faire efficacement de nombreux traitements
  - Au point qu'elles sont parfois considérées comme une science à part entière !

Expressions régulières - Exemples

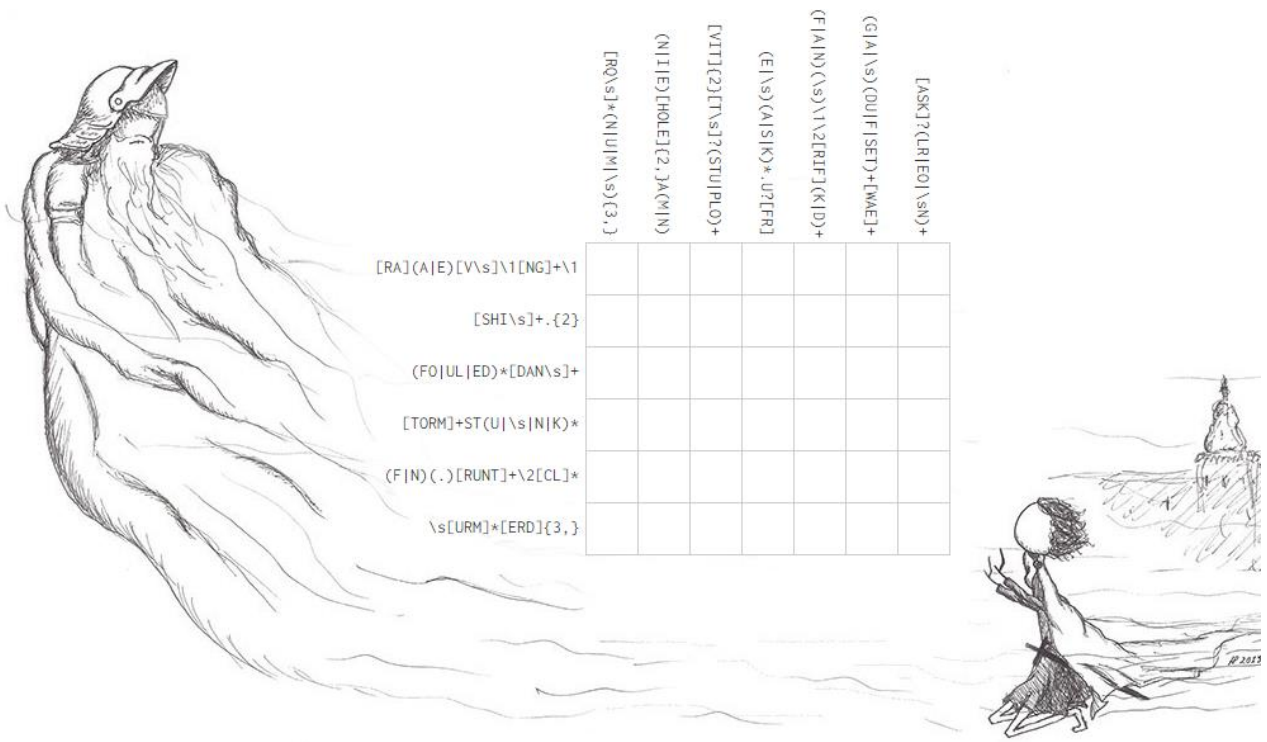
- Code postal français
  - 5 caractères de type "chiffre"
  - Expression régulière : `\d{5}`

- Email correctement formaté
  - Format attendu `email@domaine.ext`
  - Expression régulière : `\w+@\w+\.\w+`
  - Vraie expression régulière (RFC 822)

REGEX CROSSWORDS

	[^SPEAK]+	EP|IP|EF
HE|LL|O+		
[PLEASE]+		

	(MA|LM)	[^MESH]+	
[LINE]+			[ISLE]+
[LAM]+			[MALE]+
	[LAME]*	[^LES]+	



	[RQ\s]*(N|U|M|\s){3,}	(N|I|E)[HOLE]{2,}A(M|N)	[VIT]{2}[T\s]?(STU|PLO)+	(E|\s)(A|S|K)*.U?[FR]	(F|A|N)(\s)\1\2[RIF](K|D)+	(G|A|\s)(OU|F|SET)+[WAE]+	[ASK]?(LR|EO|\sN)+
[RA](A|E)[V\s]\1[NG]+\1							
[SHI\s]+.{2}							
(FO|UL|ED)*[DAN\s]+							
[TORM]+ST(U|\s|N|K)*							
(F|N)(.)[RUNT]+\2[CL]*							
\s[URM]*[ERD]{3,}							

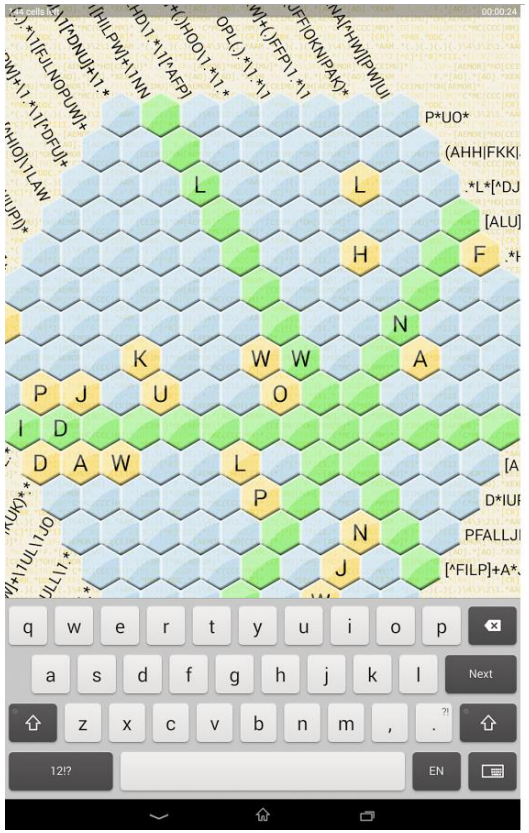
http://regexcrossword.com/

Exercices

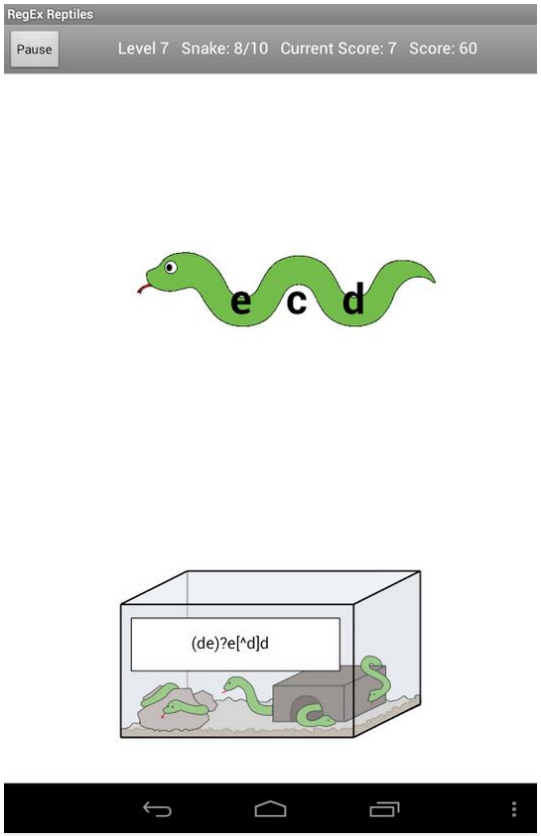
- **Exercice 1** : A quoi correspondent les expressions rationnelles suivantes sur l'alphabet $\Sigma = \{a, b, c\}$.
  1. $(a + b + c)^\star c$
  2. $(aa + bb + cc)^\star$
  3. $a^\star b^\star c^\star$

- **Exercice 2** : Peut-on trouver une expression rationnelle pour décrire les langages suivants sur l'alphabet $\Sigma = \{a, b, c\}$?
  1. Mots qui contiennent au moins un a, un b et un c
  2. Mots qui commencent et se terminent par la même lettre
  3. Mots de longueur 3 qui contiennent un nombre pair de a
  4. Mots qui contiennent un nombre pair de a et un nombre impair de b

POUR PASSER LE TEMPS…



Regex Xword



Regex Reptiles

Automates finis déterministes

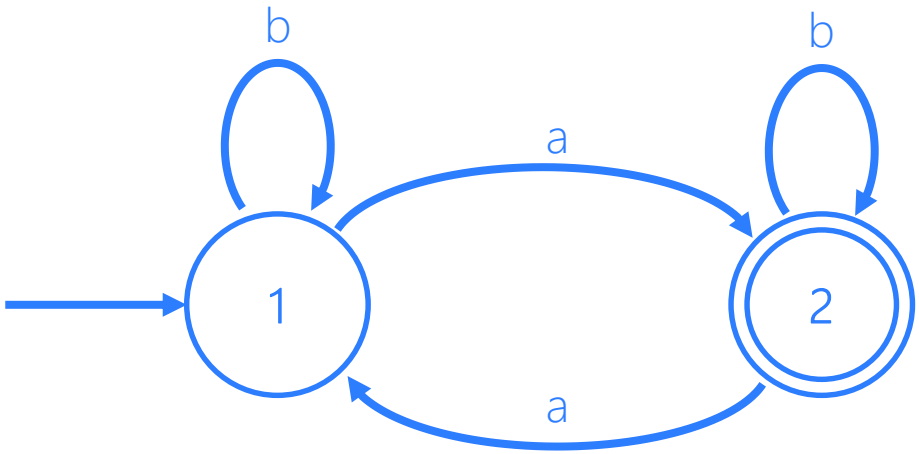
Définition formelle

- Un automate fini déterministe est défini par le quintuplet $\mathcal{A} = (\Sigma, Q, q_0, F, \delta)$
  - Σ est l'alphabet (ensemble fini de symboles)
  - Q est un ensemble fini d'états
  - $q_0 \in Q$ est l'état initial
  - $F \subset Q$ est l'ensemble des états finaux
  - δ est une fonction de $Q \times \Sigma$ dans Q, appelée fonction de transition

- Moins formellement,
  - On a plusieurs états possibles
  - On passe d'un état à un autre grâce à un symbole
  - On va chercher un chemin à partir de l'état initial pour arriver dans un état final

Représentation graphique

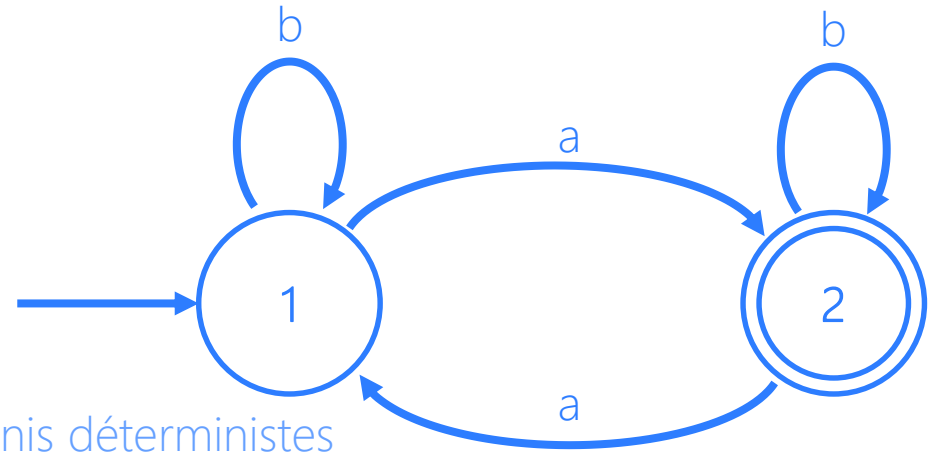
- Un automate est généralement représenté de la façon suivante
  - Les états sont représentés par des ronds
  - L'état initial est pointé par une flèche
  - Les états finaux ont un double contour
  - La fonction de transition est indiquée par des flèches
  - Ces flèches sont étiquetées par les symboles de l'alphabet



Mot reconnu par un automate

- On dit qu'un automate $\mathcal{A}$ reconnait le mot u si et seulement si :
  - En partant de l'état initial q_0
  - On épèle un-à-un les symboles qui composent u (dans l'ordre)
  - Pour chaque symbole, on suit la transition correspondante (on change généralement d'état)
  - On arrive dans l'un des états finaux

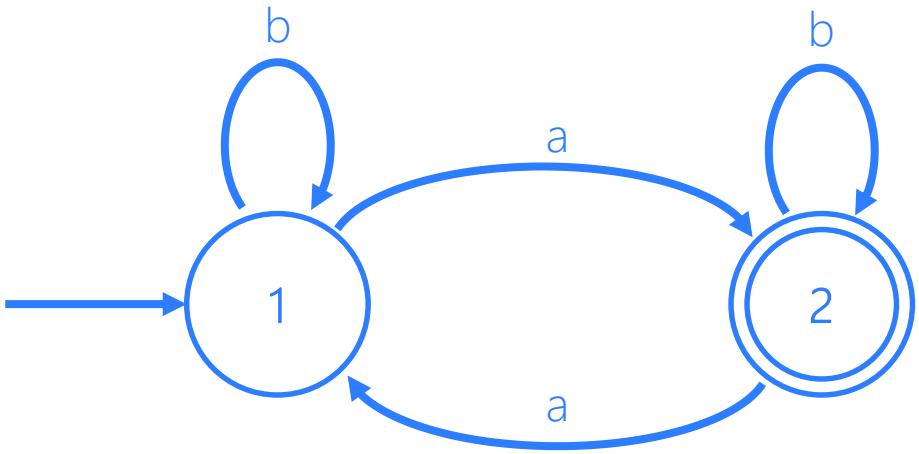
- Remarque : On parle de calcul réussi.



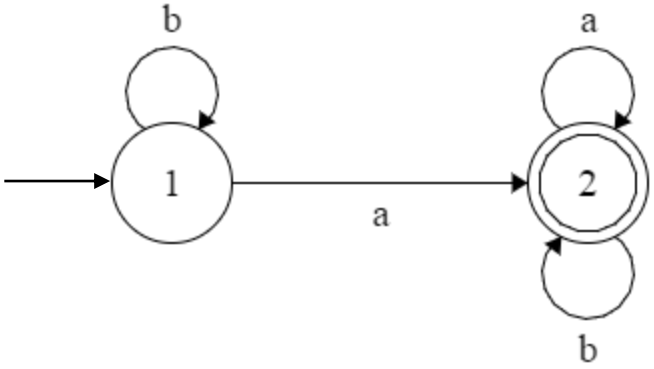
LANGAGES RECONNAISSABLES

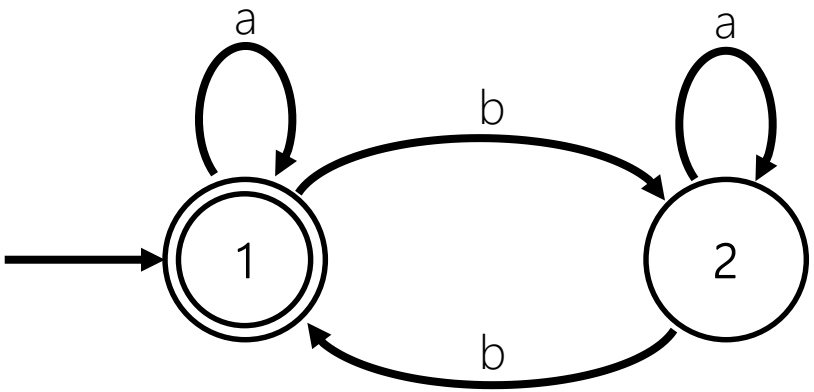
- On dit qu'un automate $\mathcal{A}$ reconnait le langage L si et seulement si :
  - L'automate $\mathcal{A}$ reconnait tous les mots qui appartiennent au langage L
  - L'automate $\mathcal{A}$ ne reconnait pas les mots qui n'appartiennent pas au langage L

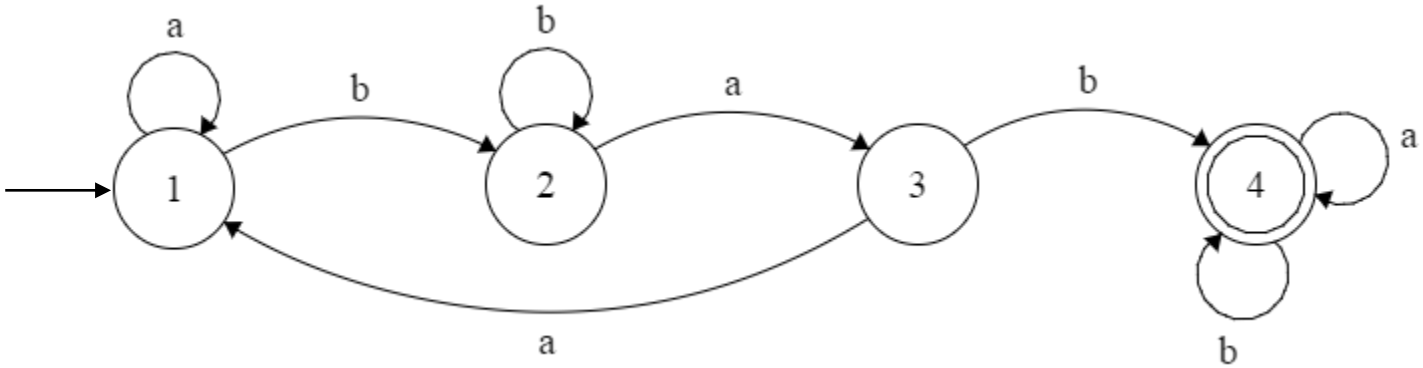
- Un tel langage est appelé reconnaissable.



QUE RECONNAIT CET AUTOMATE ?







A VOUS DE JOUER !

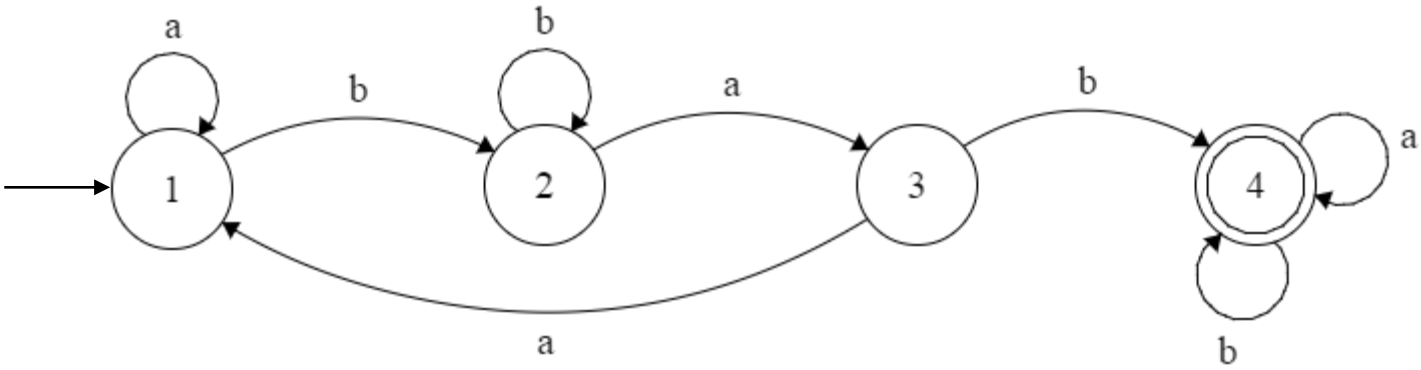
- **Exercice 1** : Dessiner les automates finis déterministes sur l'alphabet $\Sigma = \{a, b\}$ reconnaissant les langages suivants :
  1. Mots qui ne commencent pas par a
  2. Mots qui contiennent un nombre impair de b
  3. Mots qui contiennent exactement 3 a

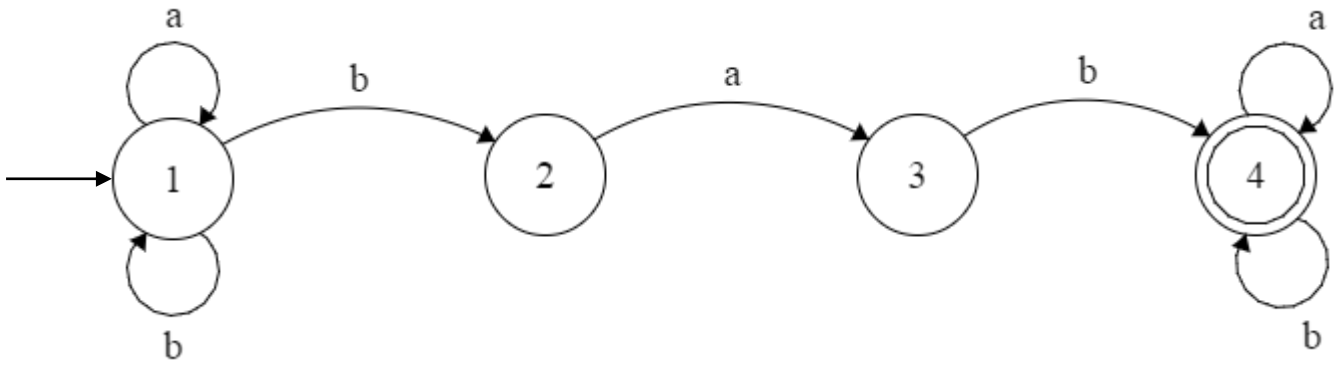
- **Exercice 2** : Dessiner les automates finis déterministes sur l'alphabet $\Sigma = \{a, b, c\}$ reconnaissant les langages suivants :
  1. Mots qui contiennent le séquence cba
  2. Mots qui contiennent au moins un a, un b, et un c

- **Exercice 3** : Peut-on dessiner un automate sur $\Sigma = \{a, b\}$ qui reconnait les mots contenant autant de a que de b ?

Automates finis non déterministes

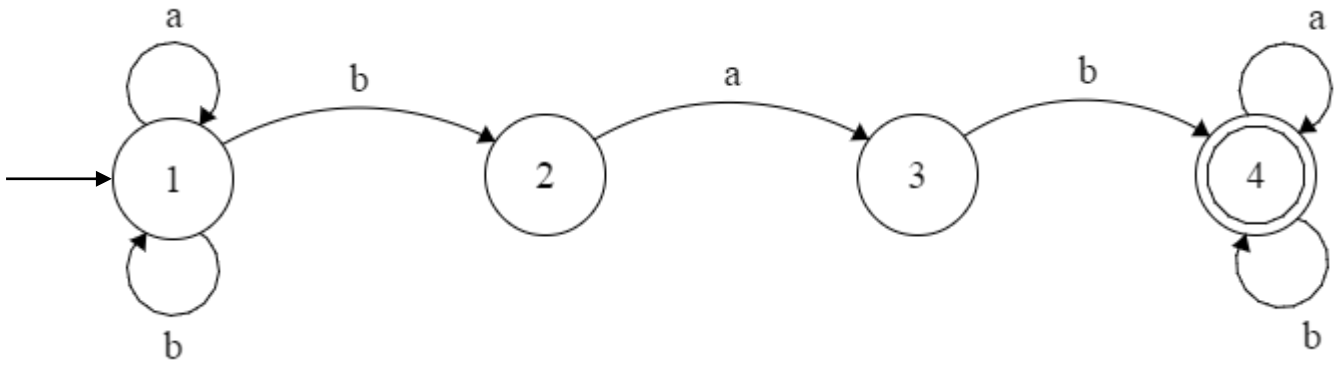
TENTATIVE DE SIMPLIFICATION





DÉFINITION

- Un automate fini non-déterministe est défini par le quintuplet $\mathcal{A} = (\Sigma, Q, I, F, \delta)$
  - Σ est l'alphabet (ensemble fini de symboles)
  - Q est un ensemble fini d'états
  - $I \subset Q$est l'ensemble des états initiaux
  - $F \subset Q$ est l'ensemble des états finaux
  - $\delta \subset Q \times \Sigma \times Q$ est une relation ; chaque triplet de δ est une transition



Déterministes et non-déterministes

- En anglais, on parle de
  - DFA : *deterministic finite automaton*
  - NFA : *nondeterministic finite automaton*

- La différence entre les automates déterministes et les automates non-déterministes est la notion de choix.

- Etat initial
  - DFA : Un seul état initial
  - NFA : Choix parmi plusieurs états initiaux possibles

- Transitions
  - DFA : Pour un état de départ et un symbole donnés, il y a toujours exactement un état de destination
  - NFA : Pour un état de départ et un symbole données, il peut exister plusieurs choix possibles pour l'état de destination

Exercice

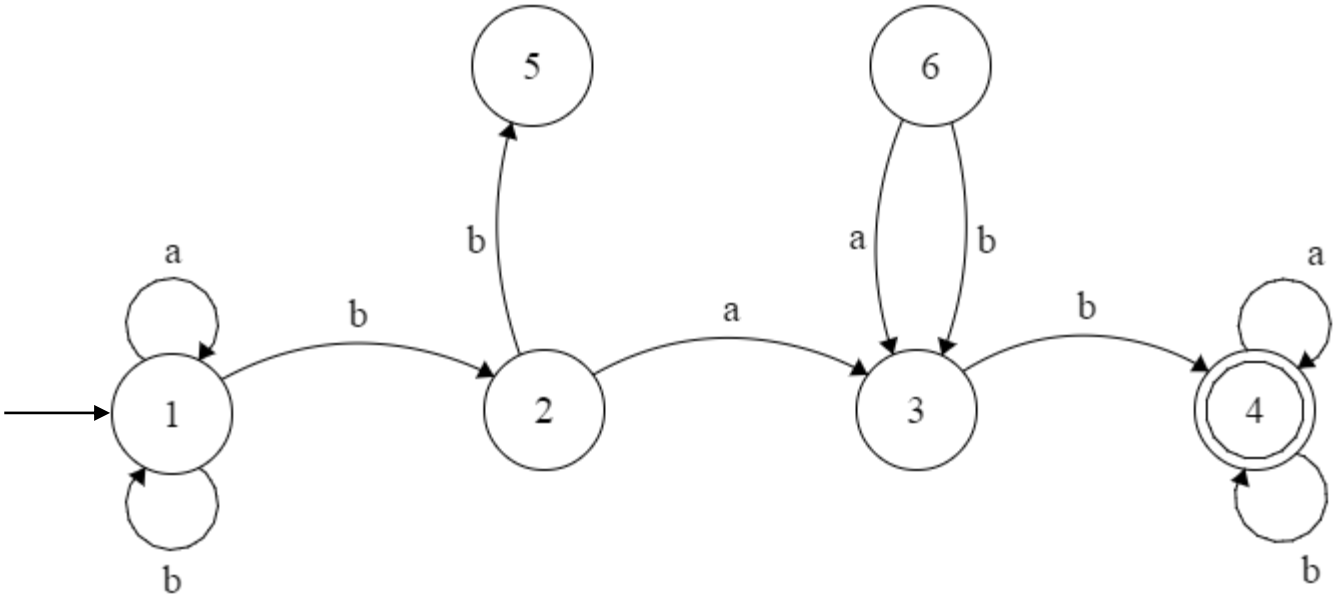
- **Exercice :** Dessiner un automate fini non-déterministe sur l'alphabet $\Sigma = \{a, b, c\}$ reconnaissant le langage suivant :

  1. Mots qui se terminent par b
  2. Mots qui contiennent exactement 3 a
  3. Mots qui contiennent le séquence cba
  4. Mots de longueur 2 ou plus, qui commencent et se terminent par la même lettre

ETATS ACCESSIBLES ET CO-ACCESSIBLES

- Un état q est dit accessible s'il existe un chemin depuis un état initial vers cet état q.

- Un état q est dit co-accessible s'il existe un chemin à partir de cet état q vers un état final.

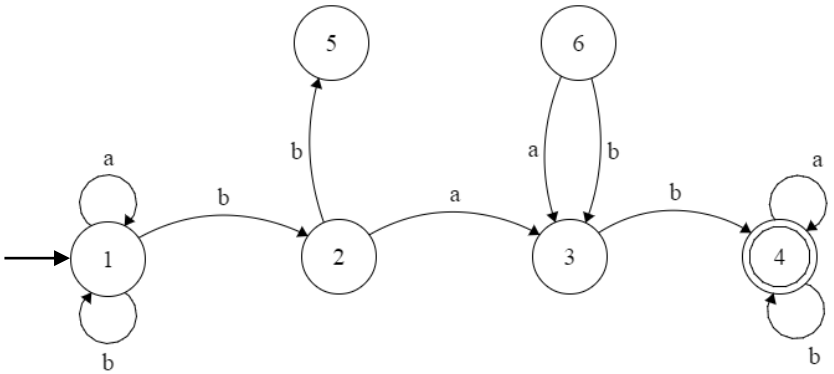


AUTOMATES ÉMONDÉS

- Un état q est dit utile s'il est à la fois accessible et co-accessible.

- Un automate dont tous les états sont utiles est dit émondé.

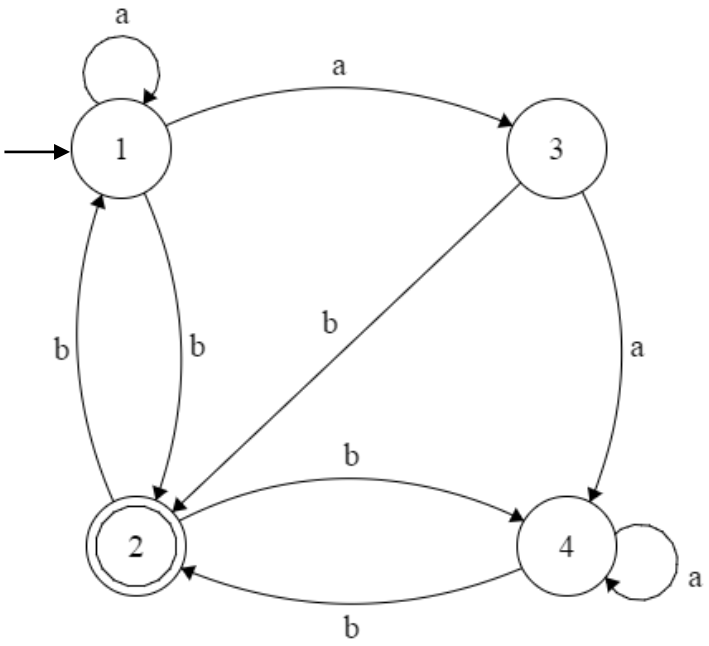
- **Théorème :** Pour tout automate fini $\mathcal{A}_1$ (déterministe ou non), il existe un automate fini émonde $\mathcal{A}_2$ reconnaissant exactement le même langage



EQUIVALENCE

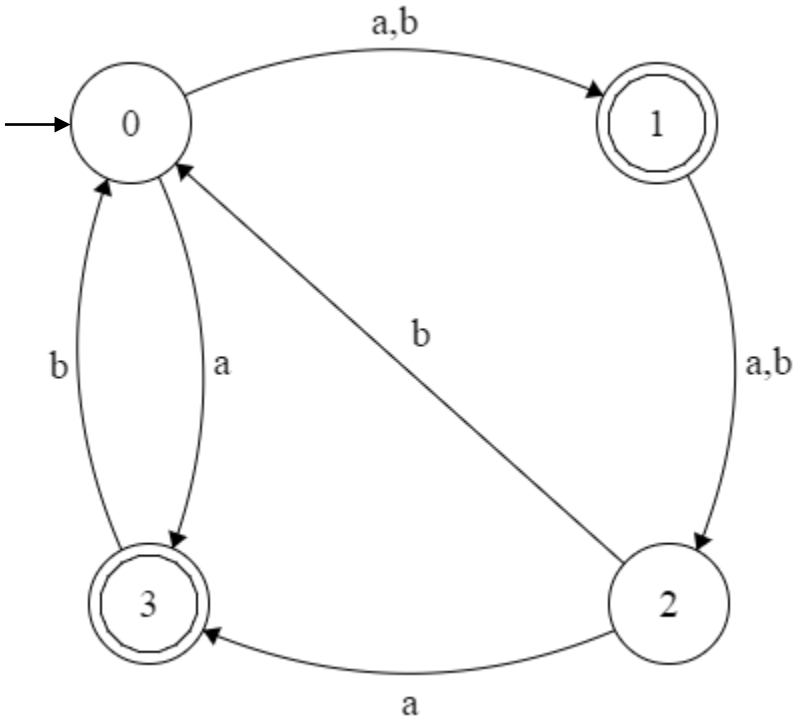
- **Théorème :** Pour tout automate fini non-déterministe $\mathcal{A}_N$, il existe un automate fini déterministe $\mathcal{A}_D$ reconnaissant exactement le même langage.

- On utilise dans la preuve la notion d' "automate des états"



EXERCICE : "DÉTERMINISATION"

- **Exercice :** Dessiner un automate fini déterministe reconnaissant le même langage que l'automate suivant :



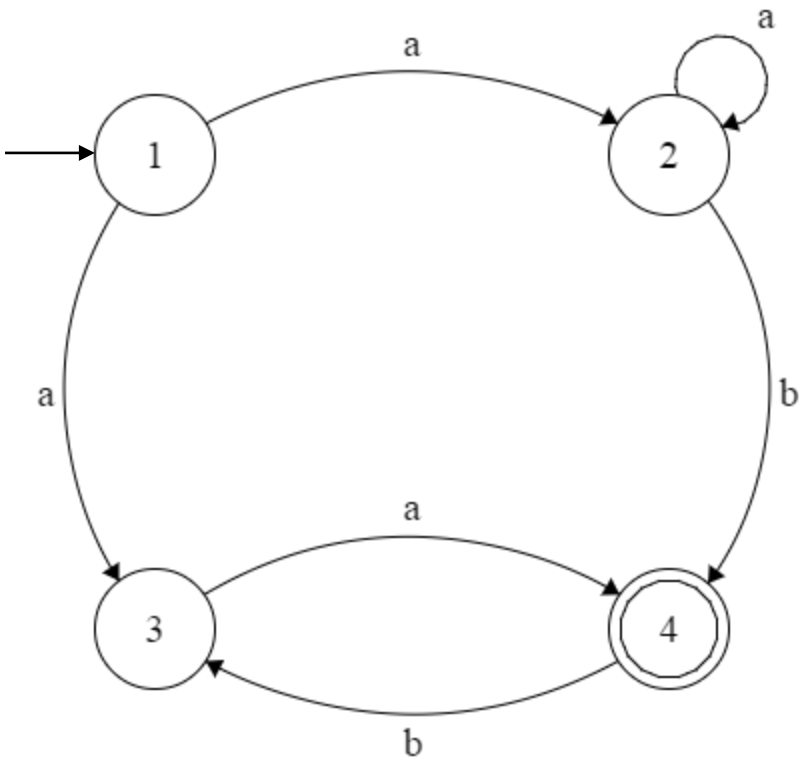
EXERCICE : RAPPELS SUR LES AUTOMATES

- **Exercice 1 :** Dessiner un automate fini non-déterministe sur l'alphabet $\Sigma = \{a, b, c\}$ reconnaissant le langage suivant :
  1. Mots qui appartiennent au langage $a^\star b^\star c^\star$
  2. Mots qui contiennent les lettres a et b dans cet ordre
  3. Mots qui contiennent un nombre pair de a et un nombre pair de b

- **Exercice 2 :** Dessiner un automate fini déterministe sur l'alphabet $\Sigma = \{a, b, c\}$ reconnaissant le langage suivant :
  1. Mots qui se terminent par ac
  2. Mots qui contiennent les lettres a et b dans cet ordre
  3. Mots qui contiennent plus de a que de b

EXERCICE : "DÉTERMINISATION"

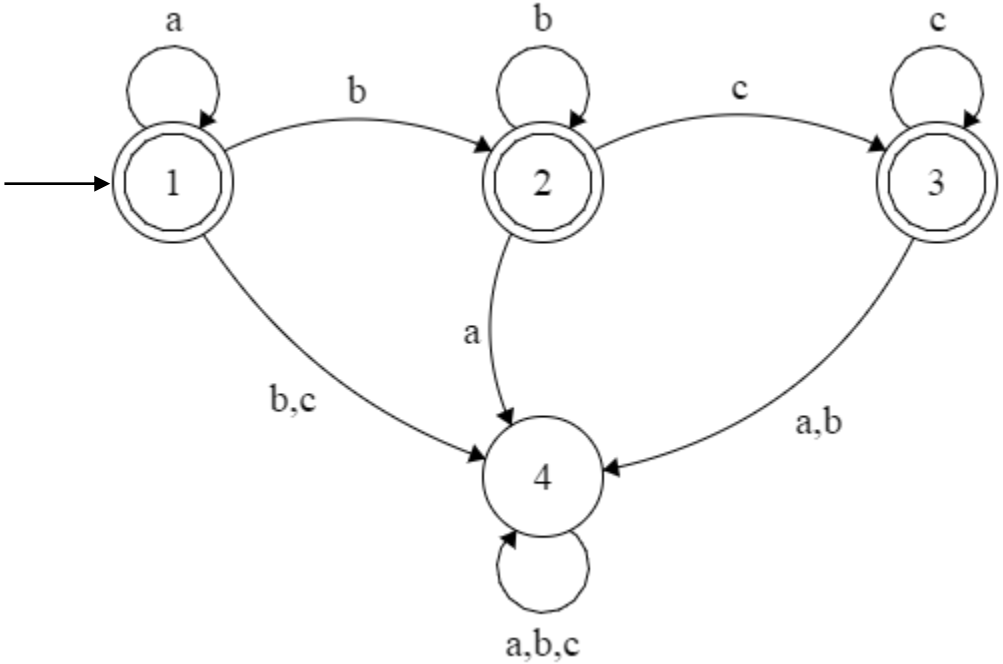
- **Exercice :** Dessiner un automate fini déterministe reconnaissant le même langage que l'automate suivant :



ε-Transitions

EXERCICE

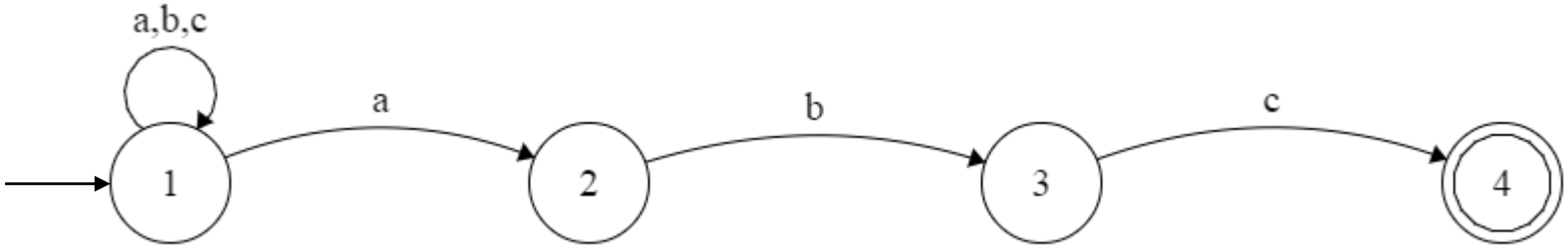
- **Exercice :** Dessiner un automate fini déterministe sur l'alphabet $\Sigma = \{a, b, c\}$ qui reconnaît le langage $a^\star b^\star c^\star$



EXERCICE : RAPPELS SUR LES AUTOMATES

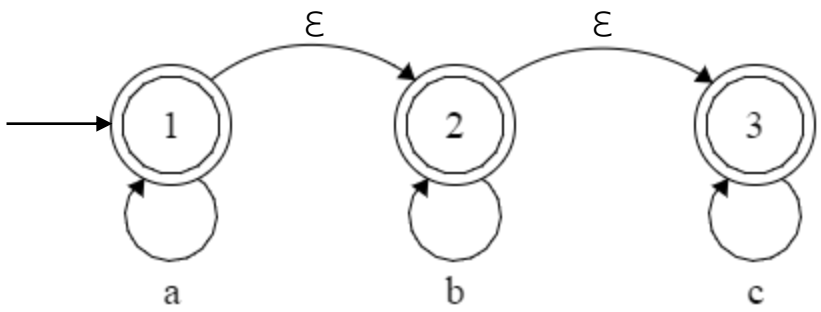
- **Exercice 1 :** Dessiner un automate fini (déterministe ou non) sur l'alphabet $\Sigma = \{a, b, c\}$ reconnaissant le langage suivant :
  1. Mots qui contient un nombre de a multiple de 3
  2. Mots qui contient un nombre de a multiple de 3, et un nombre de b non multiple de 3
  3. Mots qui contiennent les séquences ab et ac
  4. Mots qui contiennent les séquences ab et bc
  5. $\mathrm{L} = \{a^n b^n \mid n \geq 0\}$

- **Exercice 2 :** Dessiner un automate fini déterministe sur l'alphabet reconnaissant le même langage que l'automate ci-dessous :



Définition

- Un automate fini à ε-transitions (ou encore à transitions spontanées) est défini par le quintuplet $\mathcal{A} = (\Sigma, Q, I, F, \delta)$
  - Σ est l'alphabet (ensemble fini de symboles)
  - Q est un ensemble fini d'états
  - $I \subset Q$est l'ensemble des états initiaux
  - $F \subset Q$ est l'ensemble des états finaux
  - $\delta \subset Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ est une relation ; chaque triplet de δ est une transition



Equivalence

- **Théorème :** Pour tout automate fini à ε-transitions $\mathcal{A}_N$, il existe un automate fini déterministe $\mathcal{A}_D$ reconnaissant exactement le même langage.

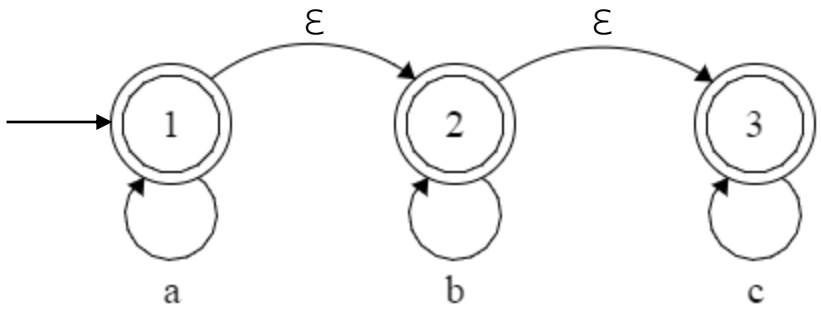
- **Méthode 1 : "ε-fermeture arrière"**
  - Principe : On cherche à se ramener à un automate fini non-déterministe en appliquant les ε-transitions "en amont" (avant chaque transition)
  - S'il existe une transition d'un état q_1 à un état q_2 étiquetée par le symbole s, alors
    - On identifie tous les états qui permettent d'accéder à l'état q_1 par une suite de transitions étiquetées par ε
    - Pour chacun de ces états q_i, on ajoute une transition de l'état q_i vers l'état q_2 étiquetée par le symbole s

EQUIVALENCE

- **Théorème :** Pour tout automate fini à ε-transitions $\mathcal{A}_N$, il existe un automate fini déterministe $\mathcal{A}_D$ reconnaissant exactement le même langage.

- **Méthode 2 : "ε-fermeture avant"**
  - Principe : On cherche à se ramener à un automate fini non-déterministe en appliquant les ε-transitions "en aval" (après chaque transition)
  - Tous état accessible à partir d'un état initial par une suite de transitions étiquetées par ε devient des états initiaux
  - S'il existe une transition d'un état q_1 à un état q_2 étiquetée par le symbole s, alors
    - On identifie tous les états accessibles à partir de l'état q_2 par une suite de transitions étiquetées par ε
    - Pour chacun de ces états q_j, on ajoute une transition de l'état q_1 vers l'état q_j et étiquetée par le symbole s

Exemple



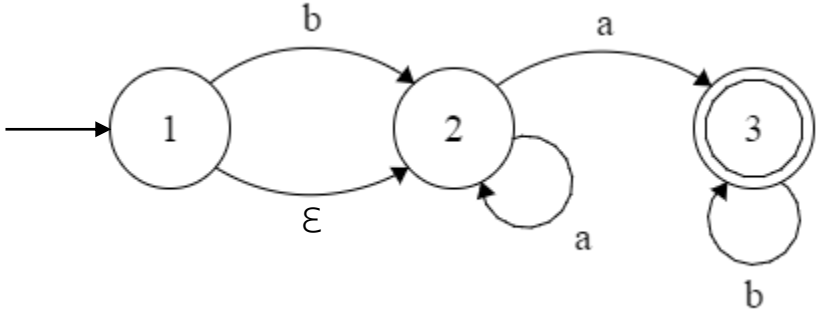
- Méthode 1 : "ε-fermeture arrière"

- Méthode 2 : "ε-fermeture arrière"

Exercices

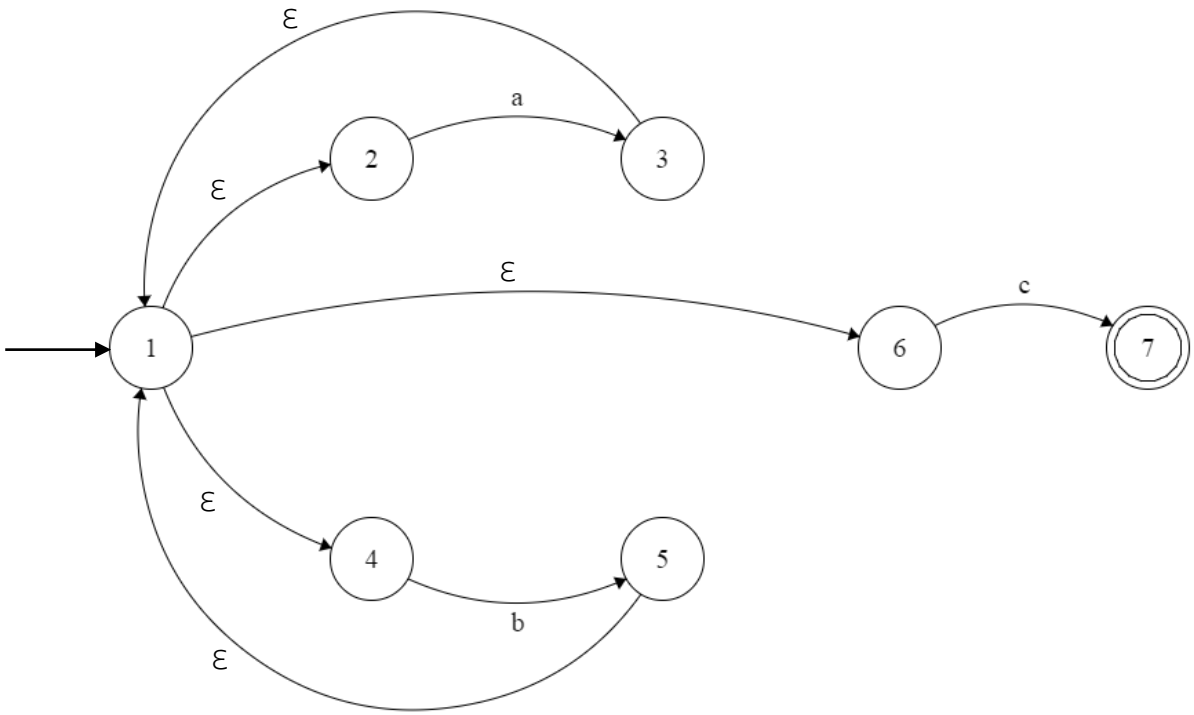
- Question 1 :
  1. Dessiner un automate fini à ε-transitions sur l'alphabet $\Sigma = \{a, b, c\}$ reconnaissant le langage décrit par l'expression $a^\star b + b a^\star$
  2. Calculer pour chaque état la liste des états accessibles par une suite de transitions étiquetées par ε
  3. Dessiner un automate fini non déterministe équivalent
  4. Dessiner un automate fini déterministe équivalent

- Question 2 : Dessiner un automate fini déterministe équivalent à l'automate ci-dessous



EXERCICES

- **Question 3 :** On considère l'automate ci-dessous

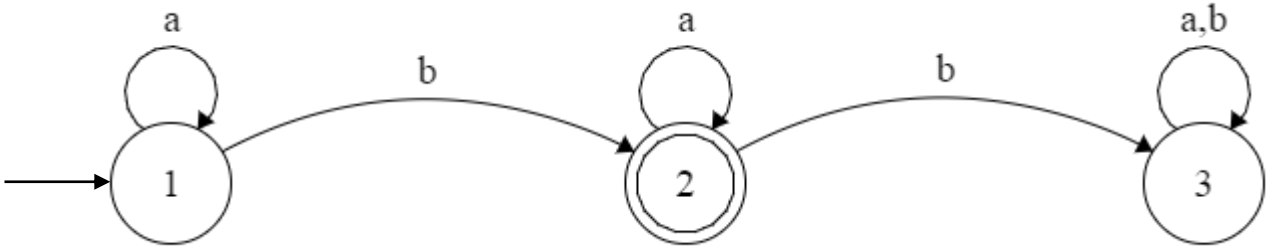


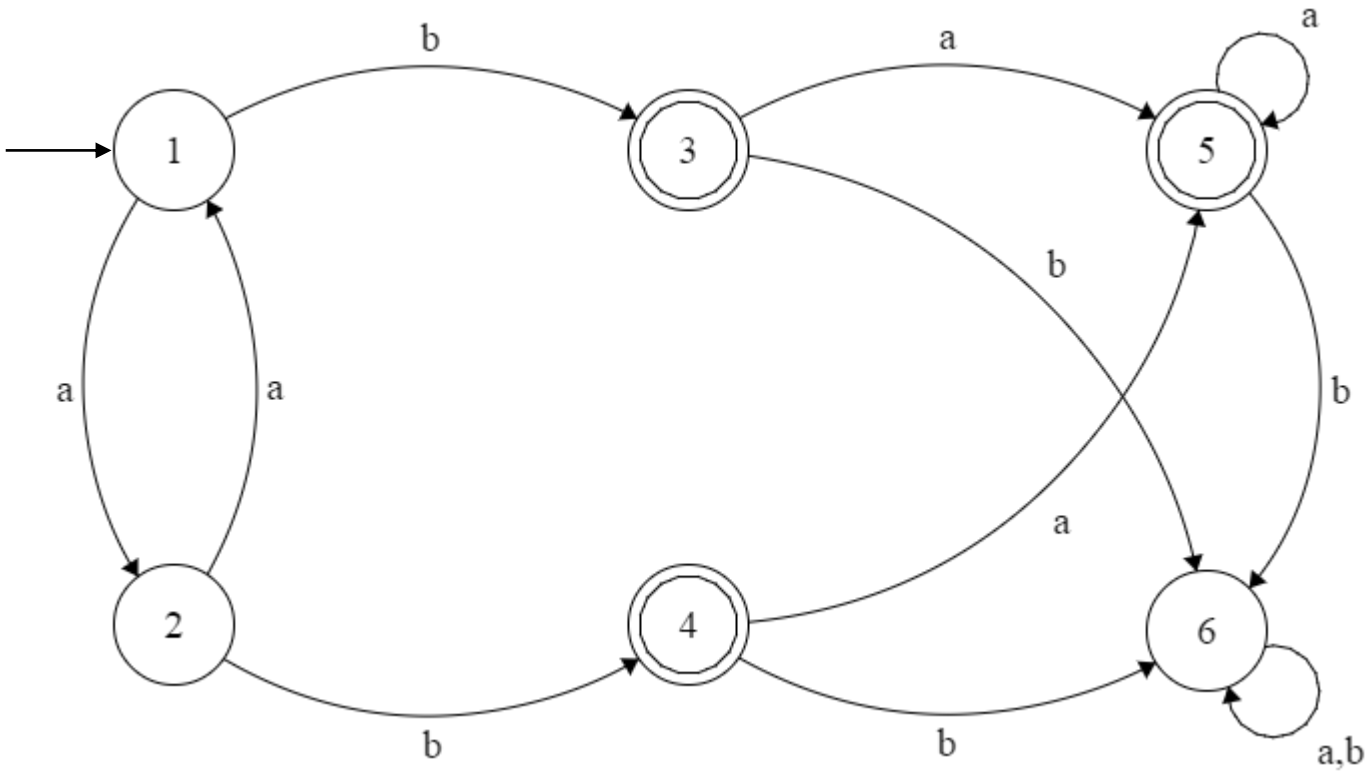
1. Construire un automate fini équivalent sans ε-transitions
2. Peut-on construire un automate fini équivalent avec moins d'états ?

Minimisation

Comparer des automates

Quels sont les langages reconnus par ces deux automates ?

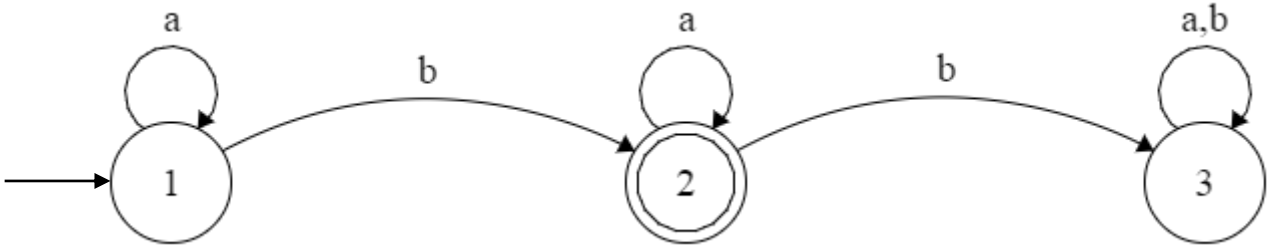




AUTOMATE MINIMAL

- **Théorème :** Pour tout langage rationnel L, il existe un plus petit automate déterministe complet reconnaissant L. Cet automate est unique à la numérotation des états près.

- **Remarque :** On parle d'automate minimal ou d'automate canonique.



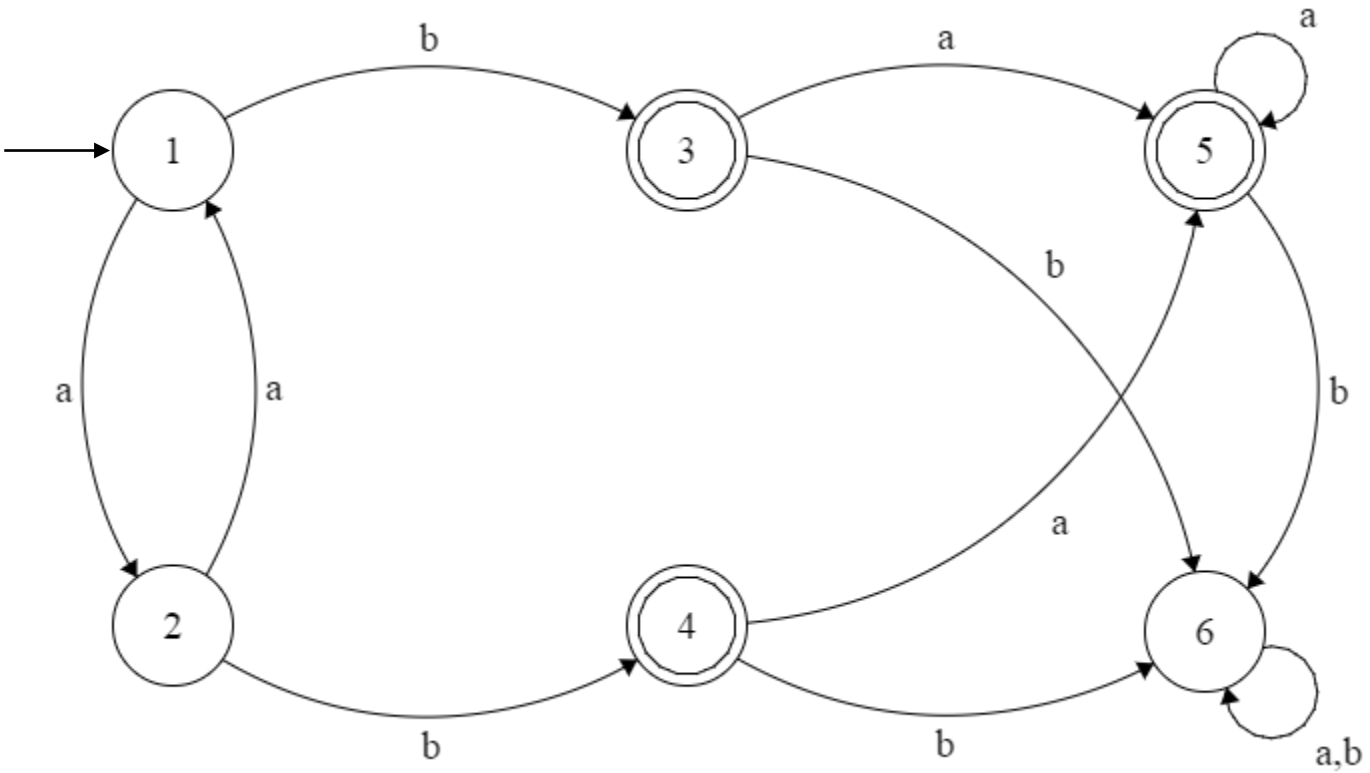
Algorithme de Moore

- **Question :** Comment construire cet automate minimal ?

- Algorithme de Moore
  - On va essayer de regrouper les états par "classe d'équivalence"
  - Approche itérative

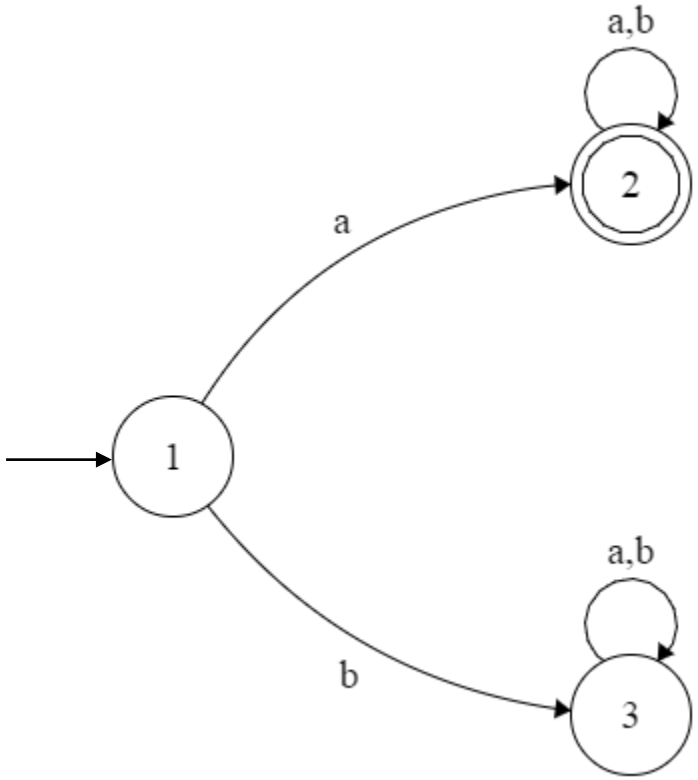
- Principe
  - On commence avec deux groupes d'états
    - L'ensemble des états finaux F
    - Les autres états de l'automate $Q \setminus F$
  - Pour chaque groupe d'états, on regarde les différents symboles étiquetant des transitions à partir d'un des états de ce groupe
    - Si un même symbole permet d'atteindre plusieurs groupes différents, il faut scinder ce groupe en plusieurs sous-groupes
  - On recommence tant qu'il reste au moins un groupe à scinder

Exemple de minimisation



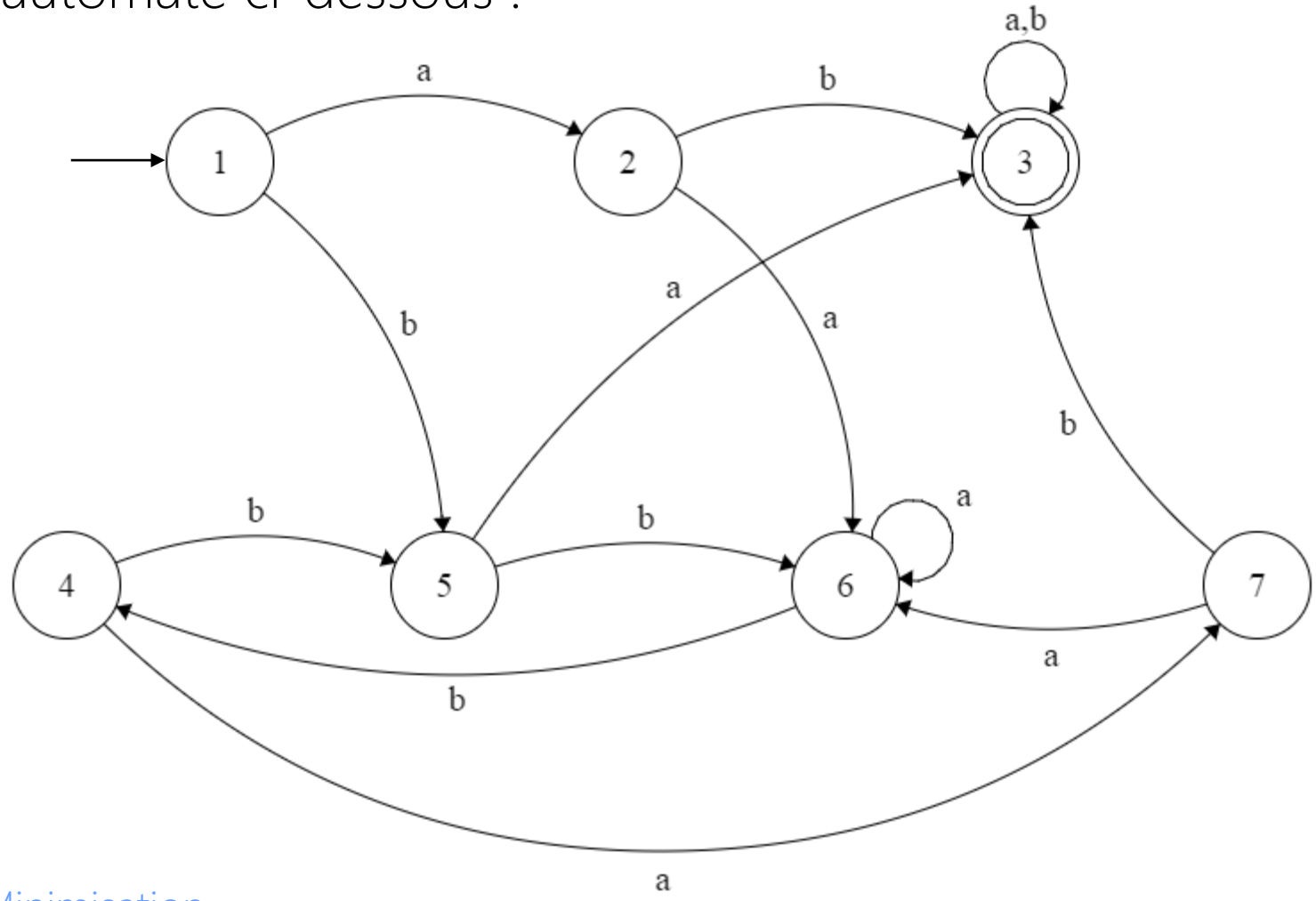
A VOTRE TOUR

- **Exercice 1 :** Dessiner l'automate minimal équivalent à l'automate ci-dessous :



A VOTRE TOUR

- **Exercice 2 :** Dessiner l'automate minimal équivalent à l'automate ci-dessous :



Automates finis et expressions rationnelles

PETIT SONDAGE

Question : Que pensez-vous des propositions suivantes ?

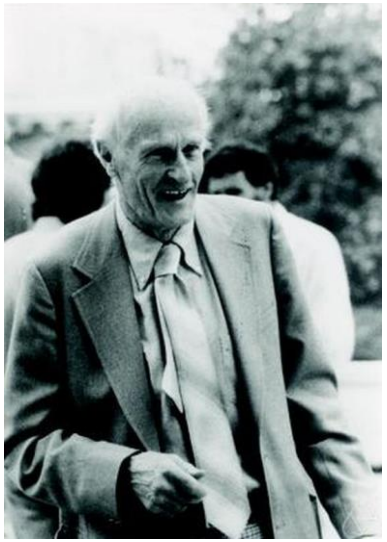
- Pour tout langage L décrit par une expression rationnelle, il existe un automate fini déterministe reconnaissant ce langage.

- Pour tout langage L reconnu par un automate fini déterministe, il existe une expression rationnelle décrivant ce langage.

Equivalence ?

- **Théorème de Kleene :** Un langage est reconnaissable (c'est-à-dire reconnu par un automate fini) si et seulement si il est rationnel (c'est-à-dire décrit par une expression rationnelle)

- Stephen Cole Kleene (1909-1994) était un mathématicien et logicien américain



RAPPEL : EXPRESSIONS RATIONNELLES

- Les expressions rationnelles sur un alphabet Σ sont définies de la façon suivante :
  - L'ensemble vide $\emptyset$ est une expression rationnelle
  - Le mot vide ε est une expression rationnelle
  - Pour tout symbole $a \in \Sigma$, a est une expression rationnelle
  - Si e_1 et e_2 sont des expressions rationnelles,
    - L'union $e_1 + e_2$ est une expression rationnelle
    - La concaténation $e_1 e_1$ est une expression rationnelle
    - L'étoile $e_1{}^\star$ est une expression rationnelle

- Remarques :
  - C'est une définition récursive
  - A chaque expression régulière peut être associé un langage

De l'expression à l'automate : objectifs

- Pour créer l'automate correspondant à une expression rationnelle, il "suffit" donc de suivre cette définition récursive.

- Il faut donc être capable de créer
  - Un automate $\mathcal{A}_{\emptyset}$ qui reconnait l'ensemble vide $\emptyset$
  - Un automate $\mathcal{A}_{\{\varepsilon\}}$ qui reconnait le langage constitué du mot vide ε
  - Pour tout symbole $a \in \Sigma$, un automate $\mathcal{A}_{\{a\}}$ qui reconnait le langage constitué du mot a
  - Si $\mathcal{A}_1$et $\mathcal{A}_2$ sont des automates finis, reconnaissant respectivement les langages L_1 et L_2,
    - Un automate $\mathcal{A}_{1+2}$ qui reconnaît la langage $L_1 \cup L_2$
    - Un automate $\mathcal{A}_{1\cdot 2}$ qui reconnaît la langage $L_1 L_2$
    - Un automate $\mathcal{A}_{1^\star}$ qui reconnaît la langage ${L_1}^\star$

De l'expression à l'automate : cas de base

- **Objectif :** Un automate $\mathcal{A}_{\emptyset}$ qui reconnait l'ensemble vide $\emptyset$

- **Objectif :** Un automate $\mathcal{A}_{\{\varepsilon\}}$ qui reconnait le langage constitué du mot vide ε

- **Objectif :** Pour tout symbole $a \in \Sigma$, un automate $\mathcal{A}_{\{a\}}$ qui reconnait le langage constitué du mot a

De l'expression à l'automate : Union

- **Objectif** : Si $\mathcal{A}_1$ et $\mathcal{A}_2$ sont des automates finis, reconnaissant respectivement les langages L_1 et L_2, un automate $\mathcal{A}_{1+2}$ qui reconnaît la langage $L_1 \cup L_2$

De l'expression à l'automate : Concaténation

- **Objectif** : Si $\mathcal{A}_1$et $\mathcal{A}_2$ sont des automates finis, reconnaissant respectivement les langages L_1 et L_2, un automate $\mathcal{A}_{1\cdot 2}$ qui reconnaît la langage $L_1 L_2$

De l'expression à l'automate : Etoile

- **Objectif :** Si $\mathcal{A}_1$ est un automate fini reconnaissant le langage L_1, un automate $\mathcal{A}_{1^\star}$ qui reconnaît la langage ${L_1}^\star$

A VOUS DE JOUER

- **Exercice 1 :** Pour chacun des langages ci-dessous, dessiner un automate fini qui reconnaît exactement ce langage :
  1. $L_1 = a(a + b + c)^\star$
  2. $L_2 = (a + bb + c)^\star$
  3. $L_3 = (a + b + c)(a + b + c)$
  4. $L_4 = L_1 L_3$
  5. $L_5 = L_1 L_2 L_3$

- **Exercice 2 :**
  1. $L_1 = a(a + b + c)^\star$

LANGAGES NON RATIONNELS

- **Question :** Existe-t-il des langages non rationnels ?

- **Exemple** : Le langage $\mathrm{L} = \{a^n b^n \mid n \geq 0\}$ est-il rationnel ?

- **Preuve :**
  - Par l'absurde, supposons que ce langage est rationnel
  - D'après le théorème de Kleene, ce langage est donc reconnaissable : il existe donc un automate fini déterministe $\mathcal{A}$ qui reconnait ce langage.
  - On note k le nombre d'états $\mathcal{A}$
  - On choisit un entier $p > k$
  - Que dire du mot $a^p b^p$?

PROCHAINE SÉANCE

Lundi 6 avril

PÂQUES