

# Cours 9: Automates finis

Olivier Bournez

bournez@lix.polytechnique.fr  
LIX, Ecole Polytechnique

INF421-a

Bases de la programmation et de l'algorithmique

# Colloquium d'**IN**formatique des **É**lèves

## Addition de Nim

## Codage, cryptographie et stéganographie élémentaires

**Daniel AUGOT**

INRIA-Saclay, École polytechnique

Jeudi 4 Novembre 2010 à 17 heures 30

*Salle de Réunion du LIX*

<http://www.enseignement.polytechnique.fr/informatique/CINE/>

# Aujourd'hui

Rappels

Déterminisation

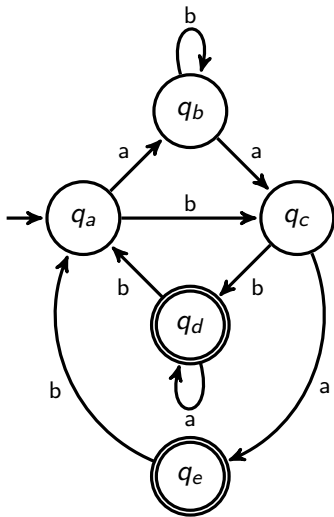
Automates et expressions régulières

Langages non-réguliers

Propriétés de clôture

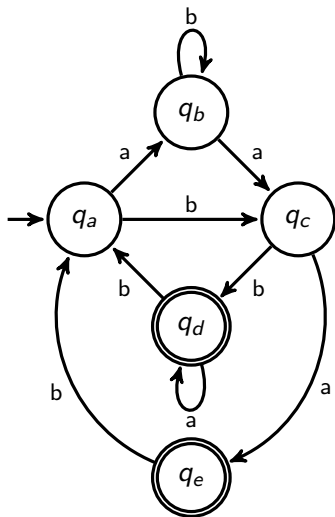
Le mot de la fin

# Automate fini déterministe (DFA)



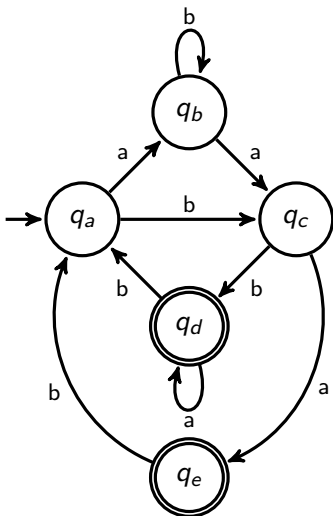
- Un automate fini déterministe est un quintuplet  $(Q, \Sigma, \delta, q_0, F)$  où
  - ▶  $\Sigma$  est un alphabet.
  - ▶  $Q$  est un ensemble fini d'états.
  - ▶  $\delta : Q \times \Sigma \rightarrow Q$  est la fonction (partielle) de transition.
  - ▶  $q_0$  est l'état initial.
  - ▶  $F \subset Q$  est un ensemble d'états finaux.

## Rappel. Langage accepté par un *DFA*: informellement



- Un automate prend en entrée un mot  $w$  et l'accepte ou le rejette:
  - ▶ On part à l'état  $q_0$
  - ▶ On lit les caractères du mot  $w$  un à un en suivant la transition correspondante (si il n'y en a pas, on se bloque = échec)
  - ▶ Lorsque tous les caractères sont lus, on accepte si l'on est sur un état final, on rejette sinon.
- Exemple:
  - ▶ Le mot *abbbab* est accepté.
  - ▶ Le mot *babba* est accepté.
  - ▶ Le mot *ababab* n'est pas accepté.

# Langage accepté par un automate: Formellement



- On introduit la *fonction  $\hat{\delta}$  de transition étendue aux mots* :

- ▶ A partir d'un état  $q$  en lisant le mot vide  $\epsilon$  on reste dans l'état  $q$ ,

$$\forall q \in Q, \hat{\delta}(q, \epsilon) = q.$$

- ▶ A partir d'un état  $q$ , en lisant le mot  $c = wa$  se terminant par  $a \in \Sigma$ , on a d'abord lu  $w$ , puis effectué la transition correspondante à  $a$

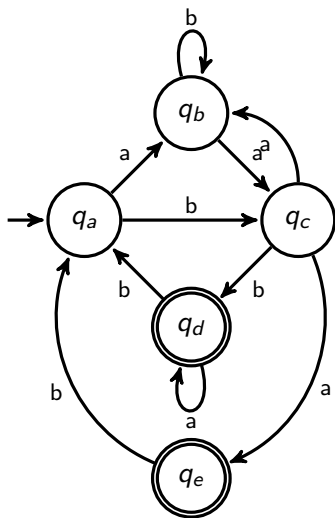
$$\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a),$$

lorsque le membre droit existe.

- Le langage  $L(A)$  accepté par  $A$  est défini par

$$L(A) = \{m \mid \hat{\delta}(q_0, m) \in F\}.$$

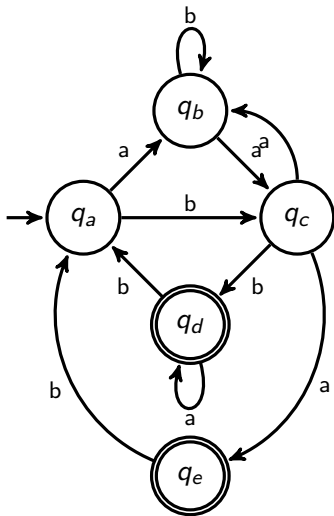
# Automate fini non-déterministe (NFA)



- Un automate fini non-déterministe est un quintuplet  $(Q, \Sigma, \delta, q_0, F)$  où
  - ▶  $\Sigma$  est un alphabet.
  - ▶  $Q$  est un ensemble fini d'états.
  - ▶  $\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q)$  est une fonction de  $Q \times \Sigma$  vers les parties  $Q$ .  $\delta$  associe à tout état  $q \in Q$ , et symbole  $c \in \Sigma$  un sous-ensemble  $\delta(q, c)$  de  $Q$ .
  - ▶  $q_0$  est l'état initial.
  - ▶  $F \subset Q$  est un ensemble d'états finaux.

# Langage accepté par un *NFA*: informellement

- Un automate prend en entrée un mot  $w$  et l'accepte ou le rejette:



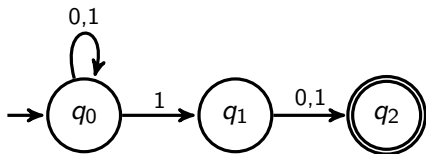
- ▶ On part à l'état  $q_0$
- ▶ On essaye de lire les caractères du mot  $w$  un à un (lorsqu'il y a plusieurs possibilités, on en choisit une/ s'il n'y en a pas, on se bloque = échec).
- ▶ S'il existe une façon de lire tous les caractères et d'arriver sur un état final, on accepte le mot.
- ▶ Sinon (il n'y a pas de façon de lire tous les caractères et d'arriver sur un état final), on rejette.

- Exemple:

- ▶ *aaabab* est accepté.
- ▶ *bab* n'est pas accepté.

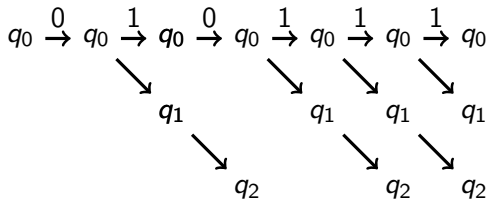
# Une façon de voir les exécutions

- Exemple:



sur  $w = 010111$ .

- Exécutions:



- On accepte un mot si et seulement s'il y a une exécution sur ce mot partant de l'état initial qui termine en un état de  $F$ .

## Formellement: langage accepté par un automate

- On introduit la *fonction  $\hat{\delta}$  de transition étendue aux mots* :

- ▶ A partir d'un état  $q$  en lisant le mot vide  $\epsilon$  on reste dans l'état  $q$ ,

$$\forall q \in Q, \hat{\delta}(q, \epsilon) = \{q\}$$

- ▶ A partir d'un état  $q$ , en lisant le mot  $m = wa$  se terminant par  $a \in \Sigma$ , on a d'abord lu  $w$ , puis effectué les transitions correspondantes à  $a$

$$\hat{\delta}(q, wa) = \bigcup_{p \in \hat{\delta}(q, w)} \delta(p, a).$$

- Le langage  $L(A)$  accepté  $A$  est défini par

$$L(A) = \{m \mid \hat{\delta}(q_0, m) \cap F \neq \emptyset\}$$

# Exercice

- Construire un automate fini déterministe qui accepte l'ensemble des mots sur l'alphabet  $\{0, 1\}$ 
  1. commençant par un 1
  2. et qui, interprétés en binaire, sont multiples de 5.
  
- Par exemple:
  - ▶ 101, 1010, et 1111 sont dans le langage.
  - ▶ 0, 100 et 111 ne le sont pas.

## Solution I/II

- L'astuce est de réaliser que lire un nouveau bit multiplie le nombre  $n$  lu jusqu'à maintenant
  - ▶ par 2 si on lit un 0;
  - ▶ par 2 et ajoute 1 si on lit un 1.
- On ne mémorise pas  $n = 5a + b$ , mais seulement le reste  $b \in \{0, 1, 2, 3, 4\}$  de la division de  $n$  par 5.
- On tabule alors en fonction de  $b$ , la nouvelle valeur de  $b$ .
- L'état  $q_i$  signifie que le nombre  $n$  vu jusque là est congru à  $i$  modulo 5:

	0	1
$\rightarrow *q_0$	$q_0$	$q_1$
$q_1$	$q_2$	$q_3$
$q_2$	$q_4$	$q_0$
$q_3$	$q_1$	$q_2$
$q_4$	$q_3$	$q_4$

## Solution II/II

- Cependant cet automate accepte des mots qui commencent par des 0.
- On ajoute un état  $s$  “initial”, et un état “mort”  $d$ .
- Si dans l'état  $s$ ,
  - ▶ on voit un 1 en premier, on se comporte comme  $q_0$ , c'est-à-dire on va en  $q_1$ .
  - ▶ on voit un 0, on n'acceptera jamais, on va en  $d$ , qu'on ne quittera jamais.

	0	1
$\rightarrow s$	$d$	$q_1$
$q_0^*$	$q_0$	$q_1$
$q_1$	$q_2$	$q_3$
$q_2$	$q_4$	$q_0$
$q_3$	$q_1$	$q_2$
$q_4$	$q_3$	$q_4$
$d$	$d$	$d$

# Aujourd'hui

Rappels

**Détermination**

Automates et expressions régulières

Langages non-réguliers

Propriétés de clôture

Le mot de la fin

## Déterminisation

- Théorème: Tout *NFA* peut être simulé par un *DFA*.

# Déterminisation

■ Théorème: Tout *NFA* peut être simulé par un *DFA*.

■ Preuve:

▶ Soit  $(Q, \Sigma, \delta, q_0, F)$  un *NFA*.

▶ Il reconnaît le même langage que le *DFA*

$$(Q', \Sigma, \delta', \{q_0\}, F')$$

avec:

- $Q' = \mathcal{P}(Q)$ :  $Q'$  est constitué de tous les sous-ensembles de  $Q$ .
- $F'$  est constitué de tous les sous-ensembles de  $Q$  avec au moins un élément commun avec  $F$ .
- Pour tout  $S \in Q'$ , et pour tout  $a \in \Sigma$ ,

$$\delta'(S, a) = \cup_{q \in S} \delta(q, a).$$

# Déterminisation

■ Théorème: Tout *NFA* peut être simulé par un *DFA*.

■ Preuve:

▶ Soit  $(Q, \Sigma, \delta, q_0, F)$  un *NFA*.

▶ Il reconnaît le même langage que le *DFA*

$$(Q', \Sigma, \delta', \{q_0\}, F')$$

avec:

- $Q' = \mathcal{P}(Q)$ :  $Q'$  est constitué de tous les sous-ensembles de  $Q$ .
- $F'$  est constitué de tous les sous-ensembles de  $Q$  avec au moins un élément commun avec  $F$ .
- Pour tout  $S \in Q'$ , et pour tout  $a \in \Sigma$ ,

$$\delta'(S, a) = \cup_{q \in S} \delta(q, a).$$

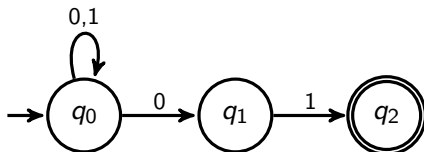
▶ Pourquoi: il est fait exactement pour que

$$\hat{\delta}'(\{q_0\}, w) = \hat{\delta}(\{q_0\}, w)$$

pour tout mot  $w$ .

## Exemple 1

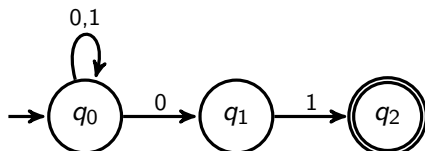
- Un *NFA*:



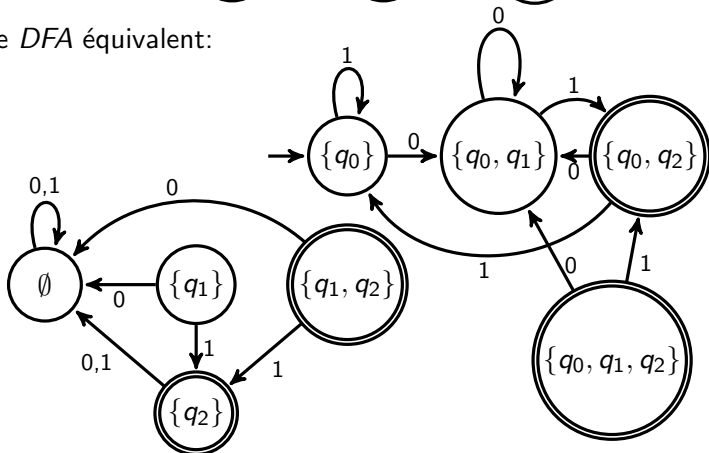
- Le *DFA* équivalent:

# Exemple 1

- Un *NFA*:

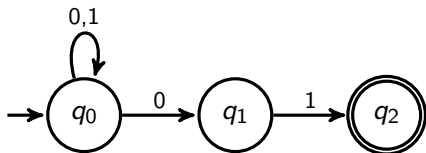


- Le *DFA* équivalent:

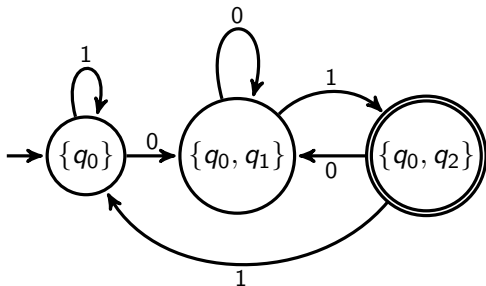


## Exemple 1: suite

- Le *NFA*:

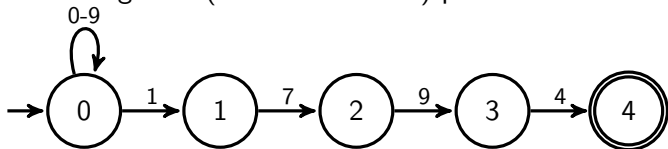


- La partie atteignable du *DFA* équivalent:



## Exemple: un digicode

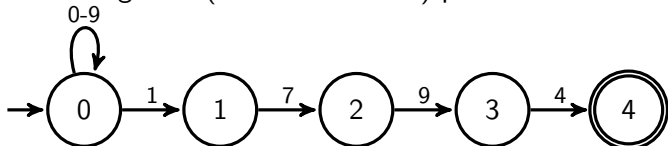
- Réaliser un digicode (code secret 1794) par un *NFA*:



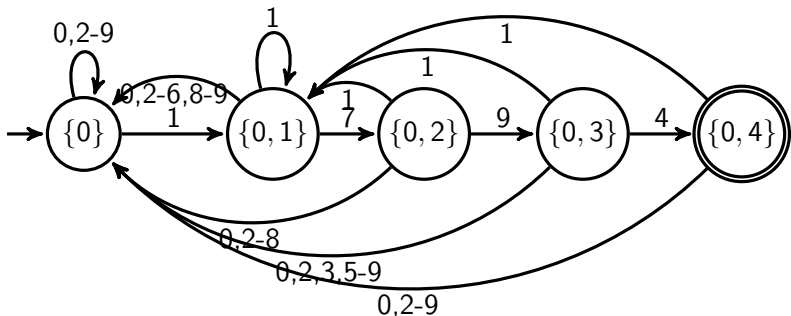
- Le *DFA* obtenu (circuit électronique correspondant):

## Exemple: un digicode

- Réaliser un digicode (code secret 1794) par un *NFA*:

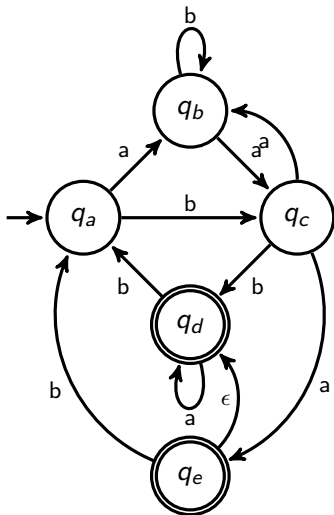


- Le *DFA* obtenu (circuit électronique correspondant):



- Tous les autres sous-ensembles de  $\{0, 1, 2, 3, 4\}$  ne sont pas atteignables: ils ne sont pas représentés ici.

# Encore une extension: automate fini non-déterministe avec $\epsilon$ -transitions (*NFA* – $\epsilon$ )



## ■ Le principe:

1. on autorise toujours plusieurs transitions avec le même symbole. (non-déterminisme)
2. mais aussi les transitions "spontanées":
  - certaines transitions peuvent être étiquetées par le mot vide  $\epsilon$ .
  - Elles peuvent être prises sans lire de lettre.

■ Exemple: le mot *baaaaa* est accepté.

## *NFA* – $\epsilon$ : formalisations

- Formalisation de la notion d'automate:
  
- Formalisation de la notion de mot reconnu:

## NFA – $\epsilon$ : formalisations

- Formalisation de la notion d'automate:

▶ ...  $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$  ...

- Formalisation de la notion de mot reconnu:

## NFA – $\epsilon$ : formalisations

- Formalisation de la notion d'automate:

▶ ...  $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$  ...

- Formalisation de la notion de mot reconnu:

▶ ...  $\hat{\delta}(q, wa) = \bigcup_{p \in \hat{\delta}(q, w)} \epsilon\text{-cl\^oture}(\delta(p, a)).$

$$\epsilon\text{-cl\^oture}(q) = \left\{ \begin{array}{l} \text{tous les \^etats joignables \^a partir de } q \\ \text{par une suite de } n \geq 0 \text{ transitions} \\ \text{\'etiquetees par } \epsilon \end{array} \right\} \dots$$

## NFA – $\epsilon$ : formalisations

- Formalisation de la notion d'automate:

▶ ...  $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$  ...

- Formalisation de la notion de mot reconnu:

▶ ...  $\hat{\delta}(q, wa) = \bigcup_{p \in \hat{\delta}(q, w)} \epsilon\text{-cl\^oture}(\delta(p, a)).$

$$\epsilon\text{-cl\^oture}(q) = \left\{ \begin{array}{l} \text{tous les \^etats joignables \^a partir de } q \\ \text{par une suite de } n \geq 0 \text{ transitions} \\ \text{\'etiquetees par } \epsilon \end{array} \right\} \dots$$

- Tout *DFA* peut \^etre vu comme un *NFA*, tout *NFA* peut \^etre vu comme un *NFA* –  $\epsilon$ .

## NFA – $\epsilon$ : formalisations

- Formalisation de la notion d'automate:

▶ ...  $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$  ...

- Formalisation de la notion de mot reconnu:

▶ ...  $\hat{\delta}(q, wa) = \bigcup_{p \in \hat{\delta}(q, w)} \epsilon\text{-cl\^oture}(\delta(p, a)).$

$$\epsilon\text{-cl\^oture}(q) = \left\{ \begin{array}{l} \text{tous les \^etats joignables \^a partir de } q \\ \text{par une suite de } n \geq 0 \text{ transitions} \\ \text{\'etiquetees par } \epsilon \end{array} \right\} \dots$$

- Tout *DFA* peut \^etre vu comme un *NFA*, tout *NFA* peut \^etre vu comme un *NFA* –  $\epsilon$ .
- Th\'eor\^eme: Tout *NFA* –  $\epsilon$  peut \^etre simul\'e par un *DFA*.

## NFA – $\epsilon$ : formalisations

- Formalisation de la notion d'automate:

- ▶ ...  $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow \mathcal{P}(Q)$  ...

- Formalisation de la notion de mot reconnu:

- ▶ ...  $\hat{\delta}(q, wa) = \bigcup_{p \in \hat{\delta}(q, w)} \epsilon\text{-cl\^oture}(\delta(p, a)).$

$$\epsilon\text{-cl\^oture}(q) = \left\{ \begin{array}{l} \text{tous les \^etats joignables \^a partir de } q \\ \text{par une suite de } n \geq 0 \text{ transitions} \\ \text{\'etiquetees par } \epsilon \end{array} \right\} \dots$$

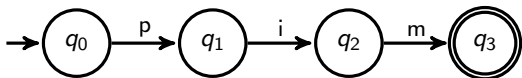
- Tout *DFA* peut \^etre vu comme un *NFA*, tout *NFA* peut \^etre vu comme un *NFA* –  $\epsilon$ .
- Th\'eor\^eme: Tout *NFA* –  $\epsilon$  peut \^etre simul\'e par un *DFA*.

- ▶ ... m\^eme principe ...

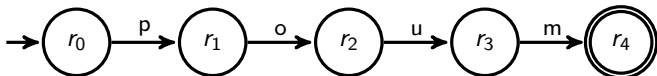
# Un intérêt: permettre de combiner directement des automates.

## ■ Exemple:

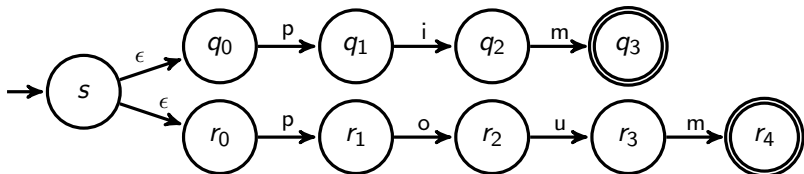
- ▶ à partir d'un automate qui reconnaît *pim*



- ▶ et d'un automate qui reconnaît *poum*



- ▶ construire un automate qui reconnaît *pim|poum* à coup de copier coller:



# Aujourd'hui

Rappels

Déterminisation

**Automates et expressions régulières**

Langages non-réguliers

Propriétés de clôture

Le mot de la fin

# Liens avec les expressions régulières

- Théorème de Kleene:

$$L(DFA) = L(NFA) = L(NFA - \epsilon) = \text{Langages réguliers.}$$

- Autrement dit: les assertions suivantes sont équivalentes.
  - ▶  $L$  est accepté par un  $DFA$ .
  - ▶  $L$  est accepté par un  $NFA$ .
  - ▶  $L$  est accepté par un  $NFA - \epsilon$ .
  - ▶  $L$  est régulier:  $L = \llbracket p \rrbracket$  pour un motif  $p$ .

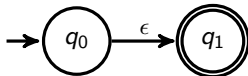
## Des expressions régulières aux automates: motifs de base

- Étant donné un motif  $p$ , on peut construire un automate qui reconnaît  $[[p]]$ .

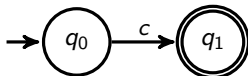
- ▶ Pour le motif  $\emptyset$ :



- ▶ Pour le motif  $\epsilon$ :



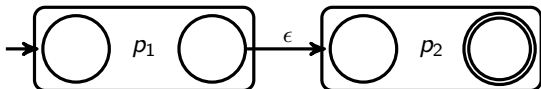
- ▶ Pour le motif  $c$ ,  $c \in \Sigma$ :



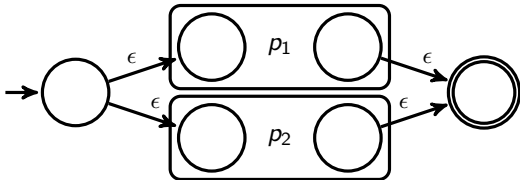
## Des expressions régulières aux automates: autres motifs

- Inductivement, si les motifs  $p_1$  et  $p_2$  correspondent à des automates,

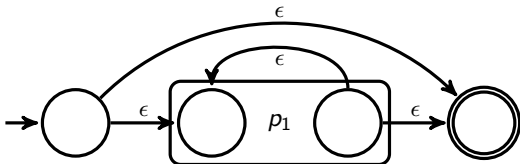
- Pour le motif  $p_1.p_2$ :



- Pour le motif  $p_1|p_2$ :



- Pour le motif  $p_1^*$ :



## Des automates aux expressions régulières: un peu d'algèbre

- Les identités  $(L_1 \cup L_2)L = L_1.L \cup L_2.L$  et  $L.(L_1 \cup L_2) = L.L_1 \cup L.L_2$  incitent à aussi noter + la disjonction:

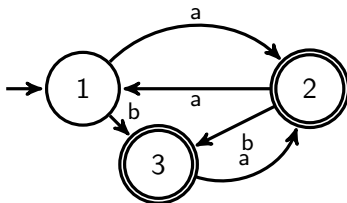
$$\begin{aligned}(L_1 + L_2)L &= L_1L + L_2L \\ L(L_1 + L_2) &= LL_1 + LL_2\end{aligned}$$

- Les identités  $\emptyset \cup L = L \cup \emptyset = L$  et  $\{\epsilon\}L = L\{\epsilon\} = L$  incitent à aussi noter 0 pour  $\emptyset$ , et 1 pour  $\{\epsilon\}$ :

$$\begin{aligned}L + 0 &= L \\ 0 + L &= L \\ 1.L &= L \\ L.1 &= L\end{aligned}$$

# Des automates aux expressions régulières: un peu d'algèbre

- Passer d'un automate à un langage...



- ... revient à résoudre un système d'équations.

$$\begin{cases} X_1 = aX_2 + bX_3 \\ X_2 = aX_1 + bX_3 + 1 \\ X_3 = aX_2 + 1 \end{cases}$$

- Cas général:

- ▶  $X_q$  est le langage constitué des mots  $w$  tel que  $\hat{\delta}(q, w) \in F$ .
- ▶  $X_q = \sum_p K_{p,q} X_p + L_q$  avec
  - $K_{p,q} = \{a \in \Sigma \mid q \in \delta(p, a)\}$ ,
  - $L_q$  vaut 0 si  $q \notin F$ , {1} sinon.

## Des automates aux expressions régulières: résoudre des équations

- Si  $K$  ne contient pas le mot vide, l'équation

$$X = KX + L$$

admet comme unique solution  $X = K^*L$ .

## Des automates aux expressions régulières: résoudre des équations

- Si  $K$  ne contient pas le mot vide, l'équation

$$X = KX + L$$

admet comme unique solution  $X = K^*L$ .

- Preuve:

- ▶  $K^*L$  est bien solution, car

$$K(K^*L) + L = (KK^*)L + L = (KK^* + 1)L = K^*L$$

# Des automates aux expressions régulières: résoudre des équations

- Si  $K$  ne contient pas le mot vide, l'équation

$$X = KX + L$$

admet comme unique solution  $X = K^*L$ .

- Preuve:

- ▶  $K^*L$  est bien solution, car

$$K(K^*L) + L = (KK^*)L + L = (KK^* + 1)L = K^*L$$

- ▶ Soit  $Y$  une autre solution, prouvons que  $X = Y$ , pour  $X = K^*L$ .

- On prouve par récurrence sur  $|w|$  que  $w \in K^*L$  implique  $w \in Y$ :  
Si  $|w| = 0$ ,  $w \in L$ , car  $\epsilon \notin K$ . Donc  $w \in KY + L = Y$ .  
Si  $|w| = n + 1$ . Si  $w \in L$ , alors  $w \in KY + L = Y$ . Sinon,  
 $w = w'w''w'''$ , avec  $w' \in K$ ,  $w'' \in K^*$ ,  $w''' \in L$ . Puisque  $w' \neq \epsilon$ ,  
 $|w''w'''| \leq n$  et par hypothèse de récurrence  $w''w''' \in Y$ , et donc  
 $w = w'w''w''' \in KY \subset KY + L = Y$ .
- Donc  $X \subset Y$ .
- De façon duale.  $Y \subset X$ .

## Des automates aux expressions régulières: résoudre des équations

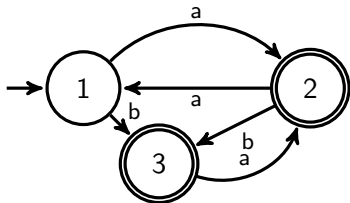
- Si  $K$  ne contient pas le mot vide, l'équation

$$X = KX + L$$

admet comme unique solution  $X = K^*L$ .

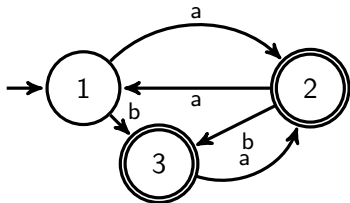
- Et donc si  $K$  et  $L$  sont réguliers,  $X$  aussi.

## Des automates aux expressions régulières: exemple



$$\begin{cases} X_1 &= aX_2 + bX_3 \\ X_2 &= aX_1 + bX_3 + 1 \\ X_3 &= aX_2 + 1 \end{cases}$$

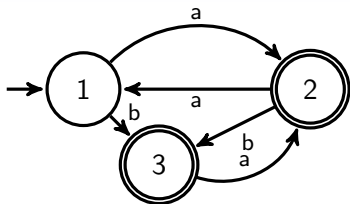
## Des automates aux expressions régulières: exemple



$$\begin{cases} X_1 &= aX_2 + bX_3 \\ X_2 &= aX_1 + bX_3 + 1 \\ X_3 &= aX_2 + 1 \end{cases}$$

- Remplaçons  $X_3$  par  $aX_2 + 1$ .

## Des automates aux expressions régulières: exemple

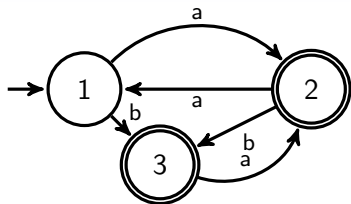


$$\begin{cases} X_1 = aX_2 + bX_3 \\ X_2 = aX_1 + bX_3 + 1 \\ X_3 = aX_2 + 1 \end{cases}$$

$$\begin{cases} X_1 = aX_2 + b(aX_2 + 1) \\ X_2 = aX_1 + b(aX_2 + 1) + 1 \\ X_3 = aX_2 + 1 \end{cases}$$

- Remplaçons  $X_3$  par  $aX_2 + 1$ .

## Des automates aux expressions régulières: exemple

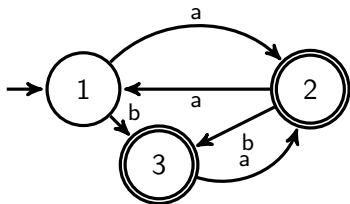


$$\begin{cases} X_1 = aX_2 + bX_3 \\ X_2 = aX_1 + bX_3 + 1 \\ X_3 = aX_2 + 1 \end{cases}$$

$$\begin{cases} X_1 = (a + ba)X_2 + b \\ X_2 = aX_1 + baX_2 + (b + 1) \\ X_3 = aX_2 + 1 \end{cases}$$

- Remplaçons  $X_3$  par  $aX_2 + 1$ .
- Remplaçons  $X_1$  par  $(a + ba)X_2 + b$ .

## Des automates aux expressions régulières: exemple

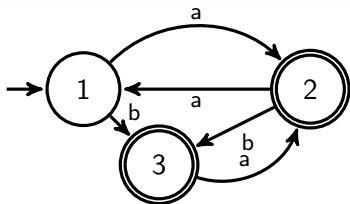


$$\begin{cases} X_1 = aX_2 + bX_3 \\ X_2 = aX_1 + bX_3 + 1 \\ X_3 = aX_2 + 1 \end{cases}$$

$$\begin{cases} X_2 = a((a + ba)X_2 + b) + baX_2 + (b + 1) \\ X_1 = (a + ba)X_2 + b \\ X_3 = aX_2 + 1 \end{cases}$$

- Remplaçons  $X_3$  par  $aX_2 + 1$ .
- Remplaçons  $X_1$  par  $(a + ba)X_2 + b$ .

## Des automates aux expressions régulières: exemple

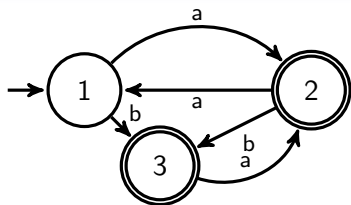


$$\begin{cases} X_1 &= aX_2 + bX_3 \\ X_2 &= aX_1 + bX_3 + 1 \\ X_3 &= aX_2 + 1 \end{cases}$$

$$\begin{cases} X_2 &= (aa + aba + ba)X_2 + (ab + b + 1) \\ X_1 &= (a + ba)X_2 + b \\ X_3 &= aX_2 + 1 \end{cases}$$

- Remplaçons  $X_3$  par  $aX_2 + 1$ .
- Remplaçons  $X_1$  par  $(a + ba)X_2 + b$ .
- Résolvons l'équation  $X_2 = (aa + aba + ba)X_2 + (ab + b + 1)$ .

## Des automates aux expressions régulières: exemple

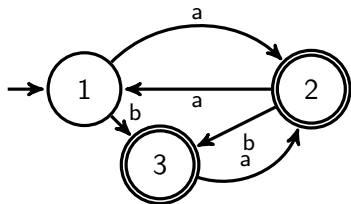


$$\begin{cases} X_1 = aX_2 + bX_3 \\ X_2 = aX_1 + bX_3 + 1 \\ X_3 = aX_2 + 1 \end{cases}$$

$$\begin{cases} X_2 = (aa + aba + ba)^*(ab + b + 1) \\ X_1 = (a + ba)X_2 + b \\ X_3 = aX_2 + 1 \end{cases}$$

- Remplaçons  $X_3$  par  $aX_2 + 1$ .
- Remplaçons  $X_1$  par  $(a + ba)X_2 + b$ .
- Résolvons l'équation  $X_2 = (aa + aba + ba)X_2 + (ab + b + 1)$ .
- Reportons.

## Des automates aux expressions régulières: exemple



$$\begin{cases} X_1 &= aX_2 + bX_3 \\ X_2 &= aX_1 + bX_3 + 1 \\ X_3 &= aX_2 + 1 \end{cases}$$

$$\begin{cases} X_2 &= (aa + aba + ba)^*(ab + b + 1) \\ X_1 &= (a + ba)(aa + aba + ba)^*(ab + b + 1) + b \\ X_3 &= a(aa + aba + ba)^*(ab + b + 1) + 1 \end{cases}$$

- Remplaçons  $X_3$  par  $aX_2 + 1$ .
- Remplaçons  $X_1$  par  $(a + ba)X_2 + b$ .
- Résolvons l'équation  $X_2 = (aa + aba + ba)X_2 + (ab + b + 1)$ .
- Reportons.
- Le langage reconnu par l'automate est donné par  $X_1$ , et donc vaut  $(a + ba)(aa + aba + ba)^*(ab + b + 1) + b$ .

# Aujourd'hui

Rappels

Déterminisation

Automates et expressions régulières

**Langages non-réguliers**

Propriétés de clôture

Le mot de la fin

## Le lemme de la “pompe”

- Pour tout langage régulier  $L$ , il existe un entier  $n$  tel que tout mot  $m$  de  $L$  de longueur  $|m| \geq n$  se décompose en  $w = xyz$  avec
  - ▶  $y \neq \epsilon$
  - ▶  $|xy| \leq n$
  - ▶ et pour tout entier  $k$ ,  $xy^kz \in L$ .

## Le lemme de la “pompe”

- Pour tout langage régulier  $L$ , il existe un entier  $n$  tel que tout mot  $m$  de  $L$  de longueur  $|m| \geq n$  se décompose en  $w = xyz$  avec
  - ▶  $y \neq \epsilon$
  - ▶  $|xy| \leq n$
  - ▶ et pour tout entier  $k$ ,  $xy^kz \in L$ .
  
- Autrement dit, si  $m$  est dans  $L$ , on peut trouver un sous mot qui peut être répété un nombre arbitraire de fois, sans changer l'appartenance à  $L$ .

## Pourquoi? I/II

- Soit  $L$  un langage régulier, et  $A$  un automate fini déterministe qui le reconnaît.
- Soit  $n$  le nombre d'états de  $A$ .
- Soit  $m$  un mot du langage  $L$  avec  $|m| \geq n$ .
- Considérons  $q_i(m)$  l'état de l'automate après avoir lu les  $i$  premières lettres de  $m$ .
- Puisque  $|m| \geq n$ , nécessairement on a du repasser au moins deux fois par le même état  $q$ 
  - ▶ (principe des tiroirs et des chaussettes/lemme des (trous de) pigeons).

## Pourquoi? II/II

- Soient  $i < j$  tels que  $q_i(m) = q_j(m) = q$ .

- Soit  $x$  constitué des  $i$  premiers caractères de  $m$ :

$$\hat{\delta}(q_0, x) = q.$$

- Soit  $y$  constitué des  $j - i$  caractères suivants de  $m$ :

$$\hat{\delta}(q_0, xy) = \hat{\delta}(q, y) = q.$$

- Soit  $z$  constitué des caractères suivants de  $m$ :

$$\hat{\delta}(q_0, xyz) = \hat{\delta}(q, z) \in F.$$

- Puisque  $\hat{\delta}(q, y) = q$ ,

$$\hat{\delta}(q, y^k) = q.$$

- On a donc

$$\hat{\delta}(q_0, xy^kz) = \hat{\delta}(q, y^kz) = \hat{\delta}(q, z) \in F.$$

# Application

- Le langage des mots sur  $\Sigma = \{a, b\}$  qui contiennent autant de  $a$  que de  $b$  n'est pas régulier.

# Application

- Le langage des mots sur  $\Sigma = \{a, b\}$  qui contiennent autant de  $a$  que de  $b$  n'est pas régulier.
- En effet

# Application

- Le langage des mots sur  $\Sigma = \{a, b\}$  qui contiennent autant de  $a$  que de  $b$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.

# Application

- Le langage des mots sur  $\Sigma = \{a, b\}$  qui contiennent autant de  $a$  que de  $b$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons le mot  $w = a^n b^n$  du langage.

# Application

- Le langage des mots sur  $\Sigma = \{a, b\}$  qui contiennent autant de  $a$  que de  $b$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons le mot  $w = a^n b^n$  du langage.
  - ▶ Supposons  $w = xyz$ , avec  $y \neq \epsilon$  et  $|xy| \leq n$ .

# Application

- Le langage des mots sur  $\Sigma = \{a, b\}$  qui contiennent autant de  $a$  que de  $b$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons le mot  $w = a^n b^n$  du langage.
  - ▶ Supposons  $w = xyz$ , avec  $y \neq \epsilon$  et  $|xy| \leq n$ .
  - ▶ Puisque  $|xy| \leq n$ ,  $x$  et  $y$  ne sont constitués que de  $a$ .

# Application

- Le langage des mots sur  $\Sigma = \{a, b\}$  qui contiennent autant de  $a$  que de  $b$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons le mot  $w = a^n b^n$  du langage.
  - ▶ Supposons  $w = xyz$ , avec  $y \neq \epsilon$  et  $|xy| \leq n$ .
  - ▶ Puisque  $|xy| \leq n$ ,  $x$  et  $y$  ne sont constitués que de  $a$ .
  - ▶ Il existe un entier  $k$  tel que  $xy^k z$  n'est pas dans le langage.

# Application

- Le langage des mots sur  $\Sigma = \{a, b\}$  qui contiennent autant de  $a$  que de  $b$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons le mot  $w = a^n b^n$  du langage.
  - ▶ Supposons  $w = xyz$ , avec  $y \neq \epsilon$  et  $|xy| \leq n$ .
  - ▶ Puisque  $|xy| \leq n$ ,  $x$  et  $y$  ne sont constitués que de  $a$ .
  - ▶ Il existe un entier  $k$  tel que  $xy^k z$  n'est pas dans le langage.
  - ▶ Contradiction.

## Autre application

- Le langage  $\{a^n b^n \mid n \geq 0\}$  n'est pas régulier.

## Autre application

- Le langage  $\{a^n b^n \mid n \geq 0\}$  n'est pas régulier.
- En effet
  - ▶ La même preuve (mot à mot) fonctionne.

## Autre application

- Le langage  $\{a^n \mid n \text{ est un entier premier}\}$  n'est pas régulier.

## Autre application

- Le langage  $\{a^n | n \text{ est un entier premier}\}$  n'est pas régulier.
- En effet

## Autre application

- Le langage  $\{a^n \mid n \text{ est un entier premier}\}$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.

## Autre application

- Le langage  $\{a^n \mid n \text{ est un entier premier}\}$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons  $p$  un nombre premier avec  $p \geq n + 2$ .

## Autre application

- Le langage  $\{a^n | n \text{ est un entier premier}\}$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons  $p$  un nombre premier avec  $p \geq n + 2$ .
  - ▶ Le mot  $w = a^p$  est dans le langage.

## Autre application

- Le langage  $\{a^n \mid n \text{ est un entier premier}\}$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons  $p$  un nombre premier avec  $p \geq n + 2$ .
  - ▶ Le mot  $w = a^p$  est dans le langage.
  - ▶ Supposons  $w = xyz$ , avec  $y \neq \epsilon$  et  $|xy| \leq n$ .

## Autre application

- Le langage  $\{a^n \mid n \text{ est un entier premier}\}$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons  $p$  un nombre premier avec  $p \geq n + 2$ .
  - ▶ Le mot  $w = a^p$  est dans le langage.
  - ▶ Supposons  $w = xyz$ , avec  $y \neq \epsilon$  et  $|xy| \leq n$ .
  - ▶ Soit  $m = |y|$ .

## Autre application

- Le langage  $\{a^n \mid n \text{ est un entier premier}\}$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons  $p$  un nombre premier avec  $p \geq n + 2$ .
  - ▶ Le mot  $w = a^p$  est dans le langage.
  - ▶ Supposons  $w = xyz$ , avec  $y \neq \epsilon$  et  $|xy| \leq n$ .
  - ▶ Soit  $m = |y|$ .
  - ▶ On doit avoir  $xy^{p-m}z$  dans le langage, de longueur  $p - m + (p - m) * m = (m + 1)(p - m)$ .

## Autre application

- Le langage  $\{a^n \mid n \text{ est un entier premier}\}$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons  $p$  un nombre premier avec  $p \geq n + 2$ .
  - ▶ Le mot  $w = a^p$  est dans le langage.
  - ▶ Supposons  $w = xyz$ , avec  $y \neq \epsilon$  et  $|xy| \leq n$ .
  - ▶ Soit  $m = |y|$ .
  - ▶ On doit avoir  $xy^{p-m}z$  dans le langage, de longueur  $p - m + (p - m) * m = (m + 1)(p - m)$ .
  - ▶ Ce nombre n'est pas premier, sauf si  $m + 1 = 1$  ou  $p - m = 1$ .

## Autre application

- Le langage  $\{a^n | n \text{ est un entier premier}\}$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons  $p$  un nombre premier avec  $p \geq n + 2$ .
  - ▶ Le mot  $w = a^p$  est dans le langage.
  - ▶ Supposons  $w = xyz$ , avec  $y \neq \epsilon$  et  $|xy| \leq n$ .
  - ▶ Soit  $m = |y|$ .
  - ▶ On doit avoir  $xy^{p-m}z$  dans le langage, de longueur  $p - m + (p - m) * m = (m + 1)(p - m)$ .
  - ▶ Ce nombre n'est pas premier, sauf si  $m + 1 = 1$  ou  $p - m = 1$ .
  - ▶ Or  $m + 1 > 1$  puisque  $m \neq 0$ , et  $p - m > 1$  puisque  $p \geq n + 2$ .

## Autre application

- Le langage  $\{a^n | n \text{ est un entier premier}\}$  n'est pas régulier.
- En effet
  - ▶ Soit  $n$  l'entier donné par le lemme.
  - ▶ Considérons  $p$  un nombre premier avec  $p \geq n + 2$ .
  - ▶ Le mot  $w = a^p$  est dans le langage.
  - ▶ Supposons  $w = xyz$ , avec  $y \neq \epsilon$  et  $|xy| \leq n$ .
  - ▶ Soit  $m = |y|$ .
  - ▶ On doit avoir  $xy^{p-m}z$  dans le langage, de longueur  $p - m + (p - m) * m = (m + 1)(p - m)$ .
  - ▶ Ce nombre n'est pas premier, sauf si  $m + 1 = 1$  ou  $p - m = 1$ .
  - ▶ Or  $m + 1 > 1$  puisque  $m \neq 0$ , et  $p - m > 1$  puisque  $p \geq n + 2$ .
  - ▶ Contradiction.

## Autre application

- L'ensemble des chaînes sur  $\Sigma = \{(, )\}$  bien parenthésées n'est pas régulier.
- etc ...

# Aujourd'hui

Rappels

Déterminisation

Automates et expressions régulières

Langages non-réguliers

**Propriétés de clôture**

Le mot de la fin

## Propriétés de clôture

1. L'union de deux langages réguliers est un langage régulier.
2. L'étoile d'un langage régulier est un langage régulier.
3. La concaténation de deux langages réguliers est un langage régulier.
4. Le complément d'un langage régulier est un langage régulier.
5. L'intersection de deux langages réguliers est un langage régulier.
6. La différence de deux langages réguliers est un langage régulier.

## Preuve

- On se donne deux langages réguliers  $L_1$  et  $L_2$ .
- $L_1$  correspond à un motif  $p_1$ ,  $L_2$  à un motif  $p_2$ .
- Alors:
  - ▶  $L_1 \cup L_2$  correspond au motif  $p_1|p_2$ .

## Preuve

- On se donne deux langages réguliers  $L_1$  et  $L_2$ .
- $L_1$  correspond à un motif  $p_1$ ,  $L_2$  à un motif  $p_2$ .
- Alors:
  - ▶  $L_1 \cup L_2$  correspond au motif  $p_1|p_2$ .
  - ▶  $L_1^*$  au motif  $p_1^*$

# Preuve

- On se donne deux langages réguliers  $L_1$  et  $L_2$ .
- $L_1$  correspond à un motif  $p_1$ ,  $L_2$  à un motif  $p_2$ .
- Alors:
  - ▶  $L_1 \cup L_2$  correspond au motif  $p_1|p_2$ .
  - ▶  $L_1^*$  au motif  $p_1^*$
  - ▶  $L_1.L_2$  au motif  $p_1.p_2$

## Preuve

- On se donne deux langages réguliers  $L_1$  et  $L_2$ .
- $L_1$  correspond à un motif  $p_1$ ,  $L_2$  à un motif  $p_2$ .
- Alors:
  - ▶  $L_1 \cup L_2$  correspond au motif  $p_1|p_2$ .
  - ▶  $L_1^*$  au motif  $p_1^*$
  - ▶  $L_1.L_2$  au motif  $p_1.p_2$
  - ▶  $L_1$  correspond à un automate fini déterministe  $(Q, \Sigma, \delta, q_0, F)$ .  
Son complémentaire  $L_1^c$  à l'automate fini déterministe  $(Q, \Sigma, \delta, q_0, Q - F)$ .

## Preuve

- On se donne deux langages réguliers  $L_1$  et  $L_2$ .
- $L_1$  correspond à un motif  $p_1$ ,  $L_2$  à un motif  $p_2$ .
- Alors:
  - ▶  $L_1 \cup L_2$  correspond au motif  $p_1|p_2$ .
  - ▶  $L_1^*$  au motif  $p_1^*$
  - ▶  $L_1.L_2$  au motif  $p_1.p_2$
  - ▶  $L_1$  correspond à un automate fini déterministe  $(Q, \Sigma, \delta, q_0, F)$ .  
Son complémentaire  $L_1^c$  à l'automate fini déterministe  $(Q, \Sigma, \delta, q_0, Q - F)$ .
  - ▶  $L_1 \cap L_2$  s'obtient par les lois de Morgan

$$(L_1 \cap L_2)^c = L_1^c \cup L_2^c$$

## Preuve

- On se donne deux langages réguliers  $L_1$  et  $L_2$ .
- $L_1$  correspond à un motif  $p_1$ ,  $L_2$  à un motif  $p_2$ .
- Alors:
  - ▶  $L_1 \cup L_2$  correspond au motif  $p_1|p_2$ .
  - ▶  $L_1^*$  au motif  $p_1^*$
  - ▶  $L_1.L_2$  au motif  $p_1.p_2$
  - ▶  $L_1$  correspond à un automate fini déterministe  $(Q, \Sigma, \delta, q_0, F)$ .  
Son complémentaire  $L_1^c$  à l'automate fini déterministe  $(Q, \Sigma, \delta, q_0, Q - F)$ .
  - ▶  $L_1 \cap L_2$  s'obtient par les lois de Morgan

$$(L_1 \cap L_2)^c = L_1^c \cup L_2^c$$

- ▶  $L_1 - L_2$  s'obtient par  $L_1 \cap L_2^c$ .

# Aujourd'hui

Rappels

Déterminisation

Automates et expressions régulières

Langages non-réguliers

Propriétés de clôture

**Le mot de la fin**

## Les points importants du cours ( $\Rightarrow$ INF – 431)

- Programmer plus et mieux.
- Des structures de données dynamiques.
- Des algorithmes sur ces structures.
- Quelques considérations de complexité.
- Automates et langages.

# La pale

- Tous les documents (poly, copie des transparents, notes) sont autorisés.
- Quand on vous demande d'écrire du code: rarement plus de 15 lignes.
- Le poly est un sur-ensemble de ce que nous avons vu en amphis.
- Réviser aussi les TPs.

# Le sondage

- c'est IMPORTANT.