

Aujourd'hui

Automates, rappels

Automates en Java

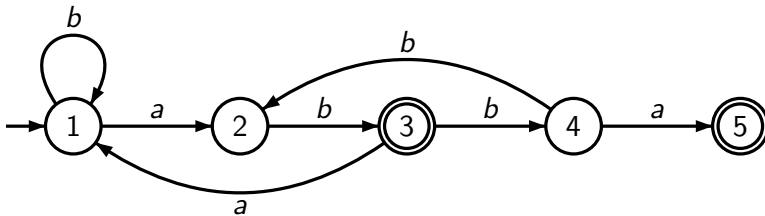
Expressions régulières, définitions

Expression régulière et automate

Tout n'est pas régulier

Conclusion

Un Automate fini déterministe



Un automate prend en entrée un mot et l'accepte ou le rejette.

INF421-a Bases de la programmation et de l'algorithmique

(Bloc 9/ 9)

Philippe Baptiste

CNRS LIX, École Polytechnique

22 octobre 2009

Automate fini non-déterministe

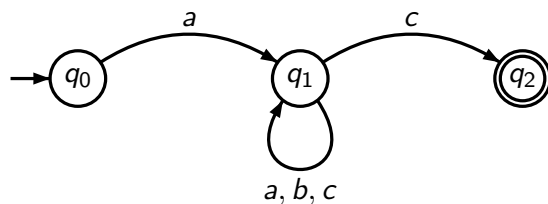
Il peut y avoir *plusieurs* transitions avec le même symbole.
Le fonctionnement d'un tel automate n'est donc PAS
totalement « DÉTERMINÉ », car on ne sait pas quel état
l'automate va choisir.

Un automate fini non-déterministe est un quintuplet :

$(Q, \Sigma, \delta, q_0, F)$

- ▶ un alphabet fini (Σ)
- ▶ un ensemble fini d'états (Q)
- ▶ une fonction de transition δ qui associe à tout état $q \in Q$ et tout symbole $s \in \Sigma$ un sous ensemble de Q noté $\delta(q, s)$.
- ▶ un état de départ (q_0)
- ▶ un ensemble d'états finaux (ou acceptant) F

Fonctionnement d'un automate fini non-déterministe



	a	b	c
→ q ₀	{q ₁ }	∅	∅
q ₁	{q ₁ }	{q ₁ }	{q ₁ , q ₂ }
* q ₂	∅	∅	∅

Java

Objectif

- ▶ Modéliser un automate fini non-déterministe (sans ϵ transition)
- ▶ Écrire un programme qui détermine si une chaîne de caractère est reconnue par l'automate.

Sans perte de généralité, nous allons supposer que l'alphabet est l'ensemble des caractères ASCII et que les états sont *numérotés* à partir de 0.

Aujourd'hui

Automates, rappels

Automates en Java

Expressions régulières, définitions

Expression régulière et automate

Tout n'est pas régulier

Conclusion

Modèle

Le modèle de données est alors très simple :

- ▶ L'état initial de l'automate est indiqué par un entier q_0 .
- ▶ La table de transition δ est un tableau bidimensionnel de listes d'entiers (la liste d'entier étant ici le moyen le plus simple de représenter un ensemble d'états). Ainsi $\delta[q][c]$ est la liste des états atteignables à partir de q en lisant le caractère c .
- ▶ L'ensemble des états finaux est une liste d'entiers.
- ▶ Enfin, le mot que l'on cherche à reconnaître est une chaîne de caractères `String mot`.

Modèle

Soit donc en Java les deux classes Liste et Automate.

```
class Liste {
    int val;
    Liste suivant;
    Liste(int v, Liste x) {
        val = v; suivant = x; }
}
class Automate {
    int q0;           // état initial
    Liste[] [] delta; // fonction de transition
    Liste finaux;    // états finaux
    Automate(int q, Liste f, Liste[] []d) {
        q0 = q;
        delta = d;
        finaux = f;
    }
}
```

Algorithme de recherche

La fonction `accepter(String mot, Automate a)` qui permet de vérifier qu'un mot `mot` est accepté par l'automate `a` appelle la fonction `static boolean accepter(String mot, Automate a, int i, int q)`.

```
static boolean accepter(String mot, Automate a) {
    return accepter(mot, a, 0, a.q0);
}
```

Algorithme de recherche

Nous aurons besoin de quelques fonctions classiques sur les listes

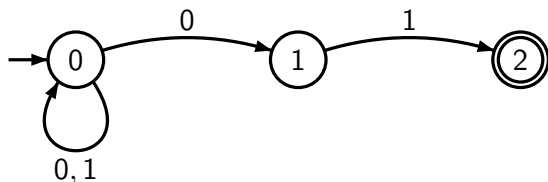
```
static int longueur(Liste x) {           // La longueur d'une liste
    if (x == null) return 0;
    else return 1 + longueur(x.suivant);
}
static int kieme(Liste x, int k) {       // Le k ème élément
    if (k == 1) return x.val;
    else return kieme(x.suivant, k-1);
}
static boolean estDans(Liste x, int v) { // Le test d'appartenance
    if (x == null) return false;
    else return x.val == v || estDans(x.suivant, v);
}
```

Algorithme de recherche

```
static boolean accepter(String mot, Automate a, int i, int q) {
    if (i == mot.length()) return Liste.estDans(a.finaux, q);
    else {
        boolean resultat = false;
        int c = mot.charAt(i); // le code ASCII du caractère courant
        for (Liste nv_q = a.delta[q][c];
             nv_q != null;
             nv_q = nv_q.suivant)
            resultat = resultat || accepter(mot, a, i+1, nv_q.val);
        return resultat;
    }
}
```

Mise en œuvre sur un automate

Considérons l'automate suivant qui accepte toutes les chaînes qui se terminent par "01".



La table associée à cet automate est alors :

	0	1
→ 0	{0, 1}	{0}
1	∅	{2}
* 2	∅	∅



Aujourd'hui

Automates, rappels

Automates en Java

Expressions régulières, définitions

Expression régulière et automate

Tout n'est pas régulier

Conclusion



Mise en œuvre sur un automate

```

public static void main(String [] arg) {
    Liste[] [] delta = new Liste[3][128];
    delta[0][((int)'0')] = new Liste(0, new Liste(1, null));
    delta[0][((int)'1')] = new Liste(0, null);
    delta[1][((int)'0')] = null;
    delta[1][((int)'1')] = new Liste(2, null);
    delta[2][((int)'0')] = null;
    delta[2][((int)'1')] = null;
    Automate a = new Automate(0,
                                new Liste(2, null),
                                delta);
    System.out.println("accepter = " + accepter(arg[0], a));
}
  
```

Remarquons que le code ASCII du caractère '0' est obtenu par (int)'0'.



Je cherche

- ▶ Comment faire pour chercher un mot dans un texte ?
- ▶ Comment faire pour chercher une expression complexe ?

```

grep -r 'xy??[0-9][0-9]' fichier.txt
egrep 'abcd|efgh' fichier.txt
egrep "([A-z]+|[0-9]+)" my.txt 1
  
```

¹displays lines containing letters in parentheses or digits in parentheses



Symboles, mots, langage

- ▶ L'alphabet Σ est un ensemble de caractères (ou symboles).
- ▶ Un mot est une suite de caractères.
- ▶ L'ensemble des mots sur Σ est noté Σ^* .
- ▶ Un *langage* est un sous-ensemble de Σ^* , c'est-à-dire un ensemble particulier de mots.
- ▶ Parmi les mots de Σ^* on distingue le mot vide noté ϵ . Le mot vide est l'unique mot de longueur zéro.

Symboles, mots, langage

- ▶ La concaténation s'étend aux ensembles de mots, on note $L_1 \cdot L_2$ le langage obtenu en concaténant tous les mots de L_1 avec les mots de L_2 .

$$L_1 \cdot L_2 = \{m_1 \cdot m_2 \mid m_1 \in L_1 \wedge m_2 \in L_2\}$$

- ▶ On note L^* le langage obtenu en concaténant les mots de L .
 - ▶ $L^0 = \{\epsilon\}$
 - ▶ $L^{n+1} = L^n \cdot L$
 - ▶ $L^* = \bigcup_{i \in \mathbb{N}} L^i$

C'est-à-dire qu'un mot m de L^* est la concaténation de n mots ($n \geq 0$) m_1, \dots, m_n , où les m_i sont tous des mots de L .

Symboles, mots, langage

- ▶ La concaténation de deux mots m_1 et m_2 est le mot obtenu en mettant m_1 à la fin de m_2 .
- ▶ On note \cdot l'opérateur de concaténation (on omet souvent cet opérateur) : $m_1 \cdot m_2$ ou $m_1 m_2$.
- ▶ La concaténation est une opération associative qui possède un élément neutre : le mot vide ϵ .
- ▶ Dans l'écriture $m = m_1 m_2$, m_1 est le *préfixe* du mot m , tandis que m_2 est le *suffixe* du mot m .

Symboles, mots, langage

Plus simple : L'union de deux langages L et M est l'ensemble des mots qui sont dans L ou dans M .

Syntaxe des expressions régulières

Les expressions régulières (ou motifs), notées p , sont définies ainsi :

- ▶ Le mot vide ϵ est une expression régulière.
- ▶ Un caractère $c \in \Sigma$ est une expression régulière.
- ▶ Si p_1 et p_2 sont des expressions régulières, alors l'alternative $p_1 \mid p_2$ est une expression régulière.
- ▶ Si p_1 et p_2 sont des expressions régulières, alors la concaténation $p_1 \cdot p_2$ est une expression régulière.
- ▶ Si p est une expression régulière, alors la répétition p^* est une expression régulière.

!! UN ARBRE!!

Syntaxe des expressions régulières

Il y a deux sortes de feuilles

- ▶ mot vide
- ▶ caractères

Deux sortes de feuilles nœuds binaires

- ▶ concaténation
- ▶ alternative

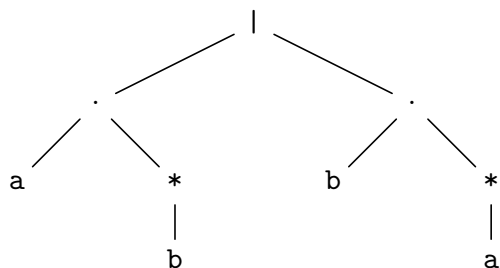
Une sorte de nœud unaire

- ▶ la répétition.

Attention, gestion des priorités : la répétition est plus prioritaire que la concaténation, elle-même plus prioritaire que l'alternative (utiliser des parenthèses)

Syntaxe des expressions régulières

Ainsi, pour $\Sigma = \{a, b, c\}$, le motif $ab^* \mid ba^*$ est à comprendre comme $((a)(b)^*) \mid ((b)(a)^*)$, ou



Sémantique des expressions régulières

- ▶ Une expression régulière a une valeur (un ensemble de mots)
- ▶ Définition inductive, qui à chaque expression régulière p associe un sous-ensemble $\llbracket p \rrbracket$ de Σ^* .

$$\begin{aligned} \llbracket \epsilon \rrbracket &= \{\epsilon\} \\ \llbracket c \rrbracket &= \{c\} \\ \llbracket p_1 \mid p_2 \rrbracket &= \llbracket p_1 \rrbracket \cup \llbracket p_2 \rrbracket \\ \llbracket p_1 \cdot p_2 \rrbracket &= \llbracket p_1 \rrbracket \cdot \llbracket p_2 \rrbracket \\ \llbracket p^* \rrbracket &= \llbracket p \rrbracket^* \end{aligned}$$

- ▶ Lorsque m appartient au langage défini par p , on dit aussi que le motif p filtre le mot m .
- ▶ Un langage qui peut être défini comme la valeur d'une expression régulière est dit *langage régulier*.

Sémantique des expressions régulières

Les représentations décimales des entiers sont un langage régulier défini par :

$$0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)^*$$

(opérateur de concaténation omis)



Les mots constitués de 0 et de 1 alternés

- ▶ $[(01)^*]$ = Les mots constitués de 0 et de 1 alternés qui commencent par 0 et finissent par 1
- ▶ $[(10)^*]$ = Les mots constitués de 0 et de 1 alternés qui commencent par 1 et finissent par 0
- ▶ $[1(01)^*]$ = Les mots constitués de 0 et de 1 alternés qui commencent par 1 et finissent par 1
- ▶ $[0(10)^*]$ = Les mots constitués de 0 et de 1 alternés qui commencent par 0 et finissent par 0

Le langage recherché est donc $[(01)^*|(10)^*|1(01)^*|0(10)^*]$

Mieux $[(\epsilon|1)(01)^*(\epsilon|0)]$



Sémantique des expressions régulières

Remarque : Un langage qui contient un nombre fini de mots est régulier.

Si $L = \{m_1, m_2, \dots, m_n\}$ est un langage fini contenant n mots, alors L est exactement décrit par l'alternative de tous les m_i .



Exercices

- ▶ $[(\epsilon|1)(00^*1)^*0^*] ???$
- ▶ $[(a^*b^*)^*aaa(a|b)^*] ???$
- ▶ $[(a|ba)^*b^*] ???$



Filtrage

La relation de filtrage $m \in \llbracket p \rrbracket$, également notée $p \preceq m$, peut être définie directement par les règles d'inférence (+ axiomes)

$$\begin{array}{c}
 \text{EMPTY} \\
 \epsilon \preceq \epsilon \\
 \\
 \text{CHAR} \\
 c \preceq c \\
 \\
 \text{SEQ} \\
 \frac{p_1 \preceq m_1 \quad p_2 \preceq m_2}{p_1 p_2 \preceq m_1 m_2} \\
 \\
 \text{ORLEFT} \\
 \frac{p_1 \preceq m}{p_1 \mid p_2 \preceq m} \\
 \\
 \text{ORRIGHT} \\
 \frac{p_2 \preceq m}{p_1 \mid p_2 \preceq m} \\
 \\
 \text{STAREMPTY} \\
 p^* \preceq \epsilon \\
 \\
 \text{STARSEQ} \\
 \frac{p \preceq m_1 \quad p^* \preceq m_2}{p^* \preceq m_1 m_2}
 \end{array}$$

Notations supplémentaires

En pratique on rencontre diverses notations qui peuvent toutes être exprimées à l'aide des constructions de base des expressions régulières. L'emploi de ces notations n'augmente pas la classe des langages réguliers, mais permet simplement d'écrire des expressions régulières plus compactes.

- ▶ Le motif optionnel $p?$ défini comme $p \mid \epsilon$.
- ▶ La répétition au moins une fois p^+ définie comme pp^* .
- ▶ Le joker, noté $.$, qui est l'alternative de tous les caractères de Σ . On a donc $\llbracket . \rrbracket = \Sigma$.

Filtrage

En enchaînant les règles on obtient une preuve effective du prédicat, appelée *arbre de dérivation*. Voici par exemple la preuve de $ab^* \mid ba^* \preceq baa$.

$$\begin{array}{c}
 \frac{\frac{a \preceq a \quad a^* \preceq \epsilon}{a^* \preceq a} \quad a \preceq a}{a^* \preceq aa} \\
 \frac{b \preceq b \quad a^* \preceq aa}{ba^* \preceq baa} \\
 \frac{ba^* \preceq baa}{ab^* \mid ba^* \preceq baa}
 \end{array}$$

Aujourd'hui

Automates, rappels

Automates en Java

Expressions régulières, définitions

Expression régulière et automate

Tout n'est pas régulier

Conclusion

Expression régulière et automate

Théorème (Kleene). Tout langage reconnu par un automate fini déterministe peut être représenté par une expression régulière, et réciproquement.

Nous allons démontrer que

- ▶ Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe (et donc par ...)
- ▶ Tout langage reconnu par un automate fini déterministe est reconnu par une expression régulière



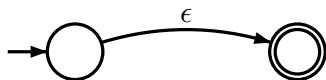
Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

- ▶ Construction par induction d'un AFN avec ϵ transition à partir des opérations élémentaires de définition des exp. rationnelles.
 - ▶ Les automates construits ont exactement un état final, pas d'arc entrant dans q_0 pas d'arc sortant de l'état final
- ▶ Base : pour \emptyset , ϵ et $c \in \Sigma$ (trivial)
- ▶ Induction



Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Pour ϵ



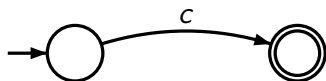
Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Pour \emptyset



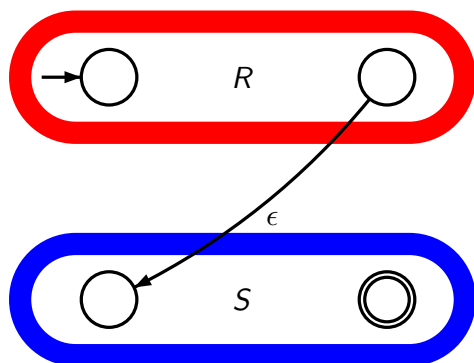
Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Pour $c \in \Sigma$



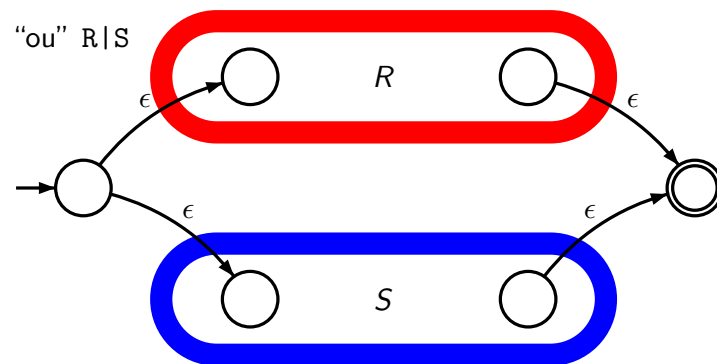
Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Induction : La concaténation RS



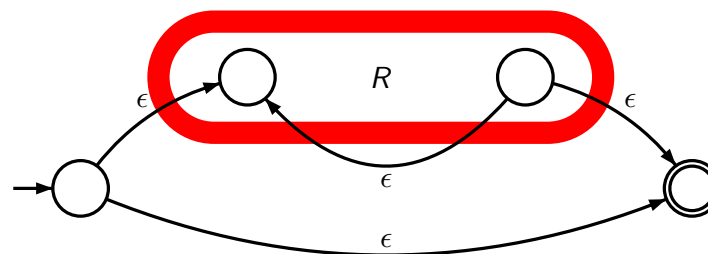
Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Induction : Le "ou" R|S



Tout langage reconnu par une expression régulière est reconnu par un automate fini non-déterministe

Induction : R^*



La réciproque : Tout langage reconnu par un automate fini déterministe est reconnu par une expression régulière

C'est un peu plus compliqué... Nous allons utiliser de la programmation dynamique.

- ▶ Partons d'un automate fini déterministe A dont les états sont $1, 2, \dots, n$. Notons $A(i, j)$ l'ensemble des symboles qui valent les arcs (i, j) .
- ▶ Si le mot m permet de passer de l'état i à l'état j dans A , alors m correspond à un chemin de i à j dans le graphe de l'automate.
- ▶ Soit R_{ij}^k l'expression régulière qui reconnaît les mots m qui correspondent à un chemin de i à j qui ne passe que par des états intermédiaires $u \leq k$

Nous allons construire toutes les expressions R_{ij}^k (induction)



Réciproque : $k = 0$

Soit m un mot qui correspond à un chemin de i à j sans sommet intermédiaire.

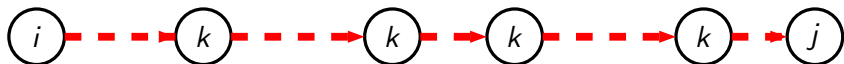
- ▶ Si $i = j$, le chemin est soit de longueur 0 (ϵ) soit c'est une boucle de i à i .
 $\Rightarrow R_{ii}^0 = (\epsilon | a_1 | \dots | a_g)$ pour $A(i, i) = \{a_1, \dots, a_g\}$
- ▶ Si $i \neq j$ et $A(i, j) = \emptyset$ alors $R_{ij}^0 = \emptyset$.
- ▶ Si $i \neq j$ et $A(i, j) \neq \emptyset$ alors $R_{ij}^0 = (a_1 | \dots | a_g)$ pour $A(i, j) = \{a_1, \dots, a_g\}$



Réciproque : Induction

Soit m un mot qui correspond à un chemin de i à j avec des sommets intermédiaires $u \leq k$.

- ▶ Si les sommets intermédiaires u sont strictement inférieurs à k alors m est reconnu par R_{ij}^{k-1}
- ▶ Sinon,



Ce qui correspond à $R_{ij}^k = (R_{ij}^{k-1} | R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1})$



Réciproque

- ▶ Attention : n^3 expressions à construire
- ▶ Les expressions générées sont abominables



Nous avons vu

Les mêmes langages sont reconnus

- ▶ par les expressions régulières
- ▶ par les automates finis déterministes
- ▶ par les automates finis non-déterministes
- ▶ par les automates finis non-déterministes avec ϵ transition

Un lemme classique (“pompage”)

Pour tout langage régulier L , il existe un entier n tel que tout mot m de L de longueur $|m|$ supérieure ou égale à n se décompose en $m = xyz$ avec

- ▶ $y \neq \epsilon$
- ▶ $|xy| \leq n$
- ▶ $\forall k \geq 0, xy^kz \in L$

En clair, les gros mots peuvent être “cassés”

Aujourd'hui

Automates, rappels

Automates en Java

Expressions régulières, définitions

Expression régulière et automate

Tout n'est pas régulier

Conclusion

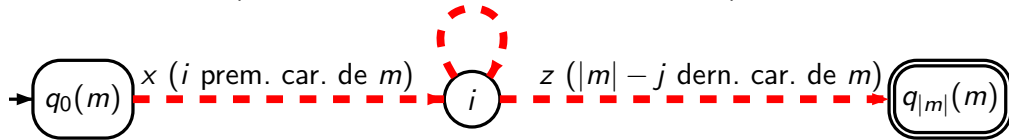
Un lemme classique (“pompage”)

- ▶ Soit L 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 $|m| \geq n$ et notons $q_i(m)$ l'état de l'automate après avoir lu les i premiers symboles de m .
- ▶ Comme $|m| \geq n$, les états $q_0(m), q_1(m), \dots, q_{|m|}(m)$ ne sont pas tous 2 à 2 distincts

Un lemme classique ("pompage")

- ▶ Soient donc i et $j > i$ les 2 premiers entiers tels que $q_i(m) = q_j(m)$

y (les car. entre les rangs $i + 1$ et j de m)



- ▶ Les mots xy^kz pour $k \geq 0$ sont donc aussi acceptés par l'automate.
- ▶ Par notre choix de i, j , nous avons $|xy| \leq n$.

Exercice

Soit $\Sigma = \{0, 1\}$ et considérons l'ensemble des mots qui contiennent autant de 1 que de 0. Le langage correspondant est-il régulier ?

Aujourd'hui

Automates, rappels

Automates en Java

Expressions régulières, définitions

Expression régulière et automate

Tout n'est pas régulier

Conclusion

Les points importants d'INF421

- ▶ Des structures de données dynamiques
- ▶ Des algorithmes sur ces structures
- ▶ Des mesures de complexité un peu plus fines
- ▶ Qques aspects de Java : L'utilisation de méthodes dynamiques, le this, les classes paramétrées
- ▶ Automates et langages

⇒ INF431

INF421, la pale

- ▶ Tous les documents sont autorisés
- ▶ Quand on vous demande d'écrire du code : rarement plus de 15 lignes (sauf XXX)
- ▶ Le poly est un sur-ensemble de ce que nous avons vu en amphi (formalisation +++)

“L'informatique”

- ▶ Domaine très très large
 - ▶ Algorithmique
 - ▶ Programmation
 - ▶ **Mais aussi** probabilité, analyse numérique, logique, algèbre, etc.
 - ▶ **Et c'est aussi** une discipline expérimentale

INF421, le sondage

C'est **IMPORTANT**