# Grammaires

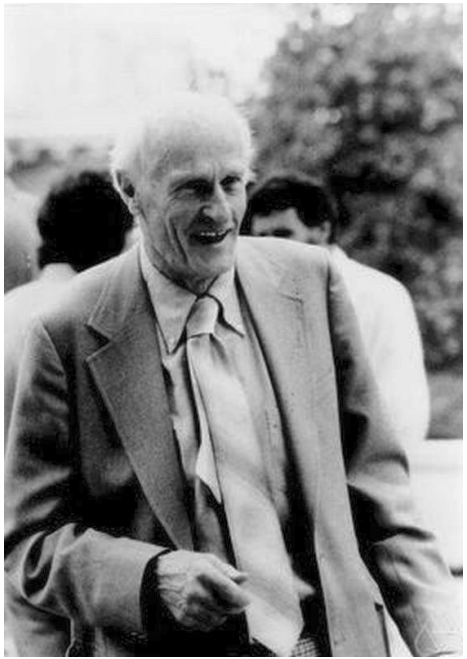
Alain Tapp

# Motivation

Introduire les expression régulière tel que décrite par Kleene. Elle permettent d'exprimer les langages réguliers en utilisant leurs fermeture sous la concaténation, l'union et l'étoile de Kleene.

Avec Python: Python RegEx

Avec Google docs: Syntax for Regular Expressions



Stephen Cole Kleene

# Expressions régulière avec variables

Une expression régulière avec variables $\mathbb{V}$ sur alphabet $\mathbf{\Sigma}$ est

- un mot A avec $A \in \mathbf{\Sigma}^*$ ou
- une variable $A \in \mathbb{V}$ ou
- la concaténation de deux expressions noté $\mathbf{A} \cdot \mathbf{B}$ ou
- l'union de deux expression est noté $\mathbf{A} \cup \mathbf{B}$ ou
- le regroupement noté (**A**) ou
- l'étoile de Kleene est noté **A***.

Priorité des opérateurs: (), * , $\cdot$ , $\cup$

# Sémantique des expressions

La sémantique des expressions régulière sur alphabet $\Sigma$ avec variables **A** et **B** suppose que les variables représentent des langages réguliers.

De plus

- $L(\mathbf{A}) = A$
- $L(\mathbf{A} \cdot \mathbf{B}) = \{x \cdot y \mid x \in L(\mathbf{A}) \text{ et } x \in L(\mathbf{A})\}$
- $L(\mathbf{A} \cup \mathbf{B}) = L(\mathbf{A}) \cup L(\mathbf{B})$
- $L(\mathbf{A}^*) = \varepsilon \text{ ou } L(\mathbf{A} \cdot \mathbf{A}^*)$

# Opération sur les automates

Étant donné les automates $A$ et $B$ on peut construire un automate C tel que $L(C)=L(A) \cup L(B)$ ce qui démontre que les langages réguliers sont fermé sous l'union (voir théorème 2.51).

Pour montrer que les langages réguliers sont fermé sous la concaténation on passe normalement par les automates non déterministes, en particulier des automates non déterministe avec des transition ε.

Les mêmes techniques permettent de montrer que les langage réguliers sont fermé sous l'étoile de Kleene.

# Notations supplémentaires

- **A B** $\stackrel{\text{def}}{=}$ **A** · **B,**
- **A** | **B** $\stackrel{\text{def}}{=}$ **A** ∪ **B**,
- **A**? $\stackrel{\text{def}}{=}$ (ε|**A**)
- $\mathbf{A}^{+}$ $\stackrel{\text{def}}{=}$ (**A**|**A***)
- **A**{n} = **A** · **A** · **A** · … · **A** (n fois)

# Grammaire Hors Contexte avec l'étoile

Une grammaire Hors Contexte avec l'étoile est un quadruplet $G=\{\mathbf{\Sigma}, V, S, R\}$ ou

- $\mathbf{\Sigma}$ est l'alphabet du langage
- $V$ est un ensemble de variables tel que $\mathbf{\Sigma} \cap V = \emptyset$
- $S$ est la variable de départ avec $S \in V$
- $R$ est un ensemble de règles de la forme $A \Rightarrow E$ ou $A \in V$ et $E$ est une expression régulière avec variables.

# Grammaire régulière

Une grammaire régulière est grammaire hors contexte avec étoile pour lequel il n'est pas possible de produire une variable un mot contenant une variable en appliquant les règles avec comme point de départ cette même variable.

$\forall A \in V$, si $A \Rightarrow^* E$ alors $|E|_A = 0$

# Exemples

$\Sigma = \{a,b\}$

- Tous les mots

  **L**=(a|b)* = $\Sigma$*
- Les mots qui commencent par a

  **L**=a(a|b)*
- Les mots qui se termine par b

  **L**=(a|b)*b
- Les mots qui on un nombre pair de a

  **L**=(b*ab*ab*)*

$\Sigma$ ={0,1,2,3,4,5,6,7,8,9,. ,E,-}

- Les chiffres non nuls
  **CP** = (1|2|3|4|5|6|7|8|9)
- Les chiffres
  **C** = 0 | **CP** = (0|1|2|3|4|5|6|7|8|9)
- Les séquences de chiffres
  **C*** = (0|1|2|3|4|5|6|7|8|9)*
- Les nombre naturels non nuls
  **P** = **CP CP*** = (1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)*
- Les nombre naturels
  **N** = 0 | **P**
- Les nombre entiers
  **Z** = (-**P) | N**
- Les nombres décimaux
  **D** = **Z** . **C**+
- Les nombres réels
  **R** = **N** | **D**
- Les nombre en notation scientifique
  **NS** = **C** . **C**+ E **Z**
- Une expression numérique
  **Z** | **R** | **NS**

# Remarques

La transformation les grammaire régulière en automate passe normalement par la construction d'un expression régulière sans variables, suivi de sa transformation en automate non déterministe puis en automate (déterministe).

En générale, l'automate équivalent à une grammaire régulière peut avoir un nombre d’états exponentielle dans la taille de la grammaire.

# Automate minimal

Étant donné une automate il est possible de produire un automate équivalent ayant un nombre minimale d'état. La complexité est de $O(s\ n \log n)$ opérations pour un automate à $n$ états sur un alphabet de $s$ symboles.

Minimisation d'un automate fini déterministe — Wikipédia

Algorithme de Hopcroft de minimisation d'un automate fini — Wikipédia

Tous les automate minimaux sont équivalent a un isomorphisme près.
(i.e. ils sont égaux si on renomme les états)

# Équivalence d’automates

En utilisant les opérations précédentes on peut facilement décider (en temps polynomial) si deux automates sont équivalents

$$A = B \quad \Leftrightarrow \quad A \subseteq B \wedge B \subseteq A$$

$$A \subseteq B \quad \Leftrightarrow \quad A \cap B^c = \emptyset$$

$$B \subseteq A \quad \Leftrightarrow \quad B \cap A^c = \emptyset$$

# Equivalence grammaire régulière

Il est possible de décider si deux grammaire régulière sont équivalents.

Ce problème est aussi difficile que tous les problèmes qui nécessitent un espace exponentiel.

Notez qu'avec un espace exponentiel on peut résoudre des problème nécessitant potentiellement un temps double exponentiel.

Un langage pas dans **REG**

$$L = \{w \mid w = a^n b^n \text{ pour } n \in \mathbb{N}\}$$

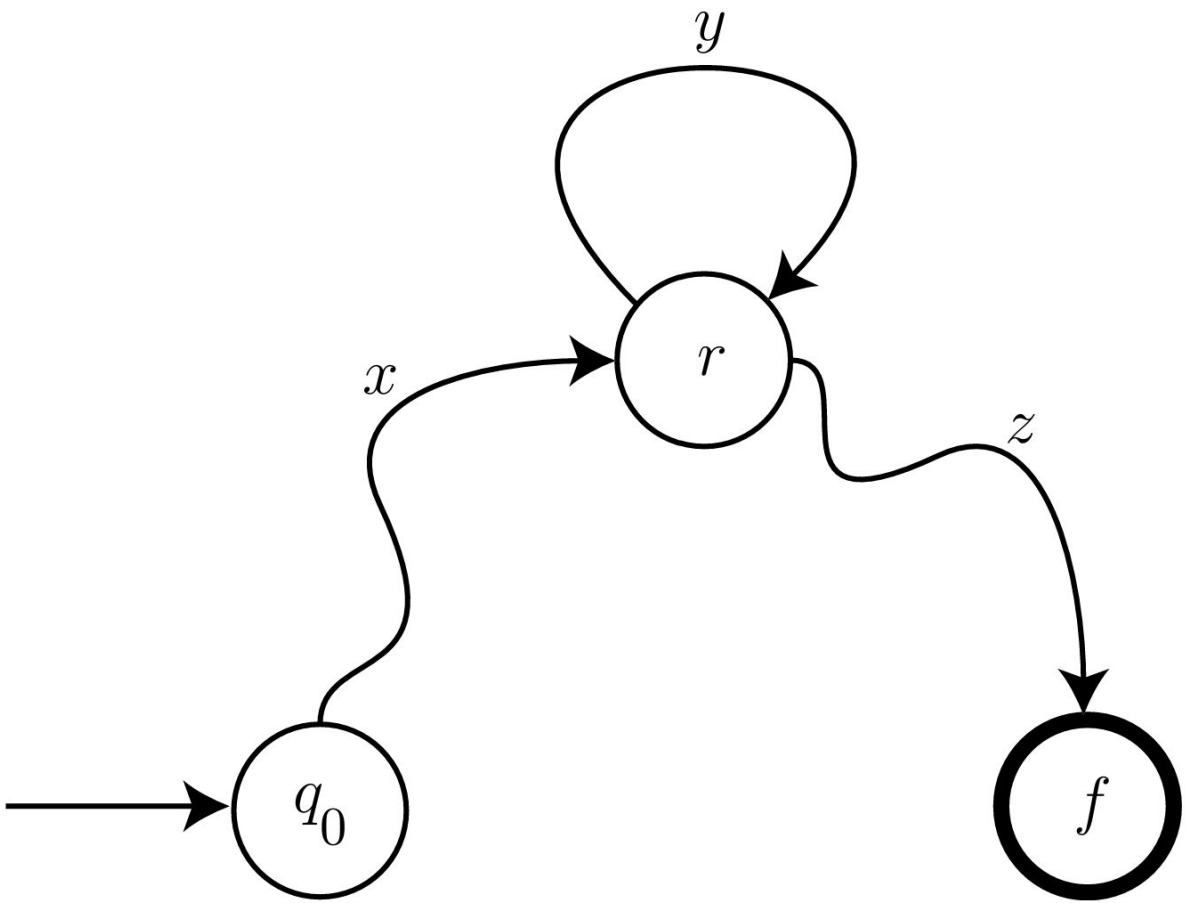
- $ab \in L$
- $aabb \in L$
- $aaabbb \in L$
- $aaaaaaabbbbbbb \in L$
- $aaaaaaabbbbbb \notin L$

Supposons que L est régulier avec 1017 états.
On a besoin de 10 bits pour représenter l'état.

$[aaaaaaabbbbbbb, q_0] \vdash^7 [bbbbbbb, q_x] \vdash^6 [b, q_y] \vdash^1 [\varepsilon , q_z]$ : $q_x, q_y \notin F, q_z \in F$

Le nombre 7 peut être codé par $q_y$ (avec moins de 10 bits).

Même chose pour n'importe quel nombre naturel.


y
r
x
z
$q_0$
f