

Théorie des langages

Damien Nouvel



Plan

1. Symboles, alphabet et mots
2. Langages générés
3. Expressions régulières

Alphabet

- ▶ **Structure mathématique**
 - **Algèbre : ensemble, opérateur** (anneau)
 - Ensemble
 - Pas de répétitions
 - L'ordre n'importe pas
 ⇒ Exemple : $\{a, c, b, c\} = \{a, b, c\}$
 - Ensemble vide : \emptyset
 - Ensemble de base : **alphabet** Σ
 - Exemples
 - Lettres : $\Sigma = \{a, b, c \dots z\}$
 - Chiffres : $\Sigma = \{0, 1, 2, 3 \dots 9\}$
 - Playstation : $\Sigma = \{haut, bas, gauche, droite, carre, rond \dots\}$
 - ...
 ⇒ Ensemble **fini** d'éléments
- ▶ Langage par **concaténation** (produit, juxtaposition)
 - Associative, non-commutative

Taille des mots

- ▶ Ensemble des **mots** formés sur Σ
 - Taille d'un mot w notée $|w|$
 - ⇒ Nombre d'éléments de Σ dans w
 - ⇒ Avec l'alphabet latin, $|langage| = 7$
 - Combinaisons possibles si $\Sigma = \{a, b\}$
 - Taille 2 : $\{aa, ab, ba, bb\}$ (4 éléments)
 - Taille 3 : $\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$ (8 éléments)
 - ⇒ Selon la taille n : 2^n
 - ⇒ Pour un alphabet de k éléments : k^n
 - ⇒ Alphabet fini et taille donnée : ensemble **fini** d'éléments
 - Il existe un **mot vide** de taille nulle noté ϵ

Opération sur les mots

► Produit

- Comme la concaténation, mais sur des langages
- Parfois notée avec le point .
- Exemple : $lang.age = langage$
- Associative, non-commutative, neutre ϵ , absorbant $\emptyset @$

► Puissance

- Itération du produit sur un élément
- Exemple : $(lang)^3 = langlanglang$

► Autre opérations

- **Préfixe** (propre) : ab est préfixe propre de $abc = ab.c$
- **Suffixe** (propre) : bc est suffixe propre de $abc = a.bc$
- **Mirroir** : $\tilde{abc} = cba$
- ...

Plan

1. Symboles, alphabet et mots
2. Langages générés
3. Expressions régulières

Opérations sur les langages

▶ Union

- **Tous** les éléments de deux ensembles
- Exemple : $\{ab, cd\} \cup \{efg, hij\} = \{ab, cd, efg, hij\}$
- Associative, commutative, neutre \emptyset

▶ Intersection

- Éléments **communs** de deux ensembles
- Exemple : $\{ab, ac, cd\} \cap \{ac, dc\} = \{ac\}$
- Associative, commutative, absorbant \emptyset (neutre Σ^*)

▶ Différence

- Tous les éléments du premier ensemble sauf ceux du second
- Exemple : $\{ab, ac, cd\} \setminus \{ac\} = \{ab, cd\}$
- Non-associative, non-commutative (neutre \emptyset , absorbant Σ^*)

▶ Complémentaire

- Tous les éléments du langage sauf ceux de l'ensemble
- Exemple : $\overline{\{b\}} = \{a, c, d \dots aa, ab, ac \dots ba, bb, bc \dots\}$

Opérations sur les langages (suite)

- ▶ **Produit**

- Concaténations possibles de mots des langages
- Exemple : $\{ab, ba\} \cdot \{cde, edc\} = \{abcde, abedc, bacde, baedc\}$
- Mêmes propriétés que pour les mots
- Distributif pour l'union (pas l'inter. $\{a, aa\} \cdot (\{b\} \cap \{ab\})$)

- ▶ **Puissance**

- Concaténations possibles des mots d'un langage
- Exemple : $\{ab, bac\}^2 = \{abab, abbac, bacab, bacbac\}$
- Mêmes propriétés que pour les mots

- ▶ **Quotient** (droit ou gauche), miroir (palindromes) ...

⇒ Pas d'intersection entre puissances : $i \neq j \Rightarrow \Sigma^i \cap \Sigma^j = \emptyset$

⇒ Le mot vide est la puissance 0 : $L^0 = \epsilon$

Langage généré (fermeture transitive)

- ▶ En théorie, pas de limite à la taille d'un mot
- ▶ **Fermeture transitive** de la concaténation
 - Union des puissances possibles
 - ⇒ $L^0 \cup L^1 \cup L^2 \cup \dots \infty$
- ▶ Étoile de **Kleene** (ou itéré, fermeture transitive)
 - Pour un alphabet, langage généré Σ^*
 - ⇒ Tous les mots possibles à partir de l'alphabet
 - Pour un langage L^*
 - ⇒ Toutes les concaténations de mots de L

Le langage comme monoïde libre

▸ Monoïde libre

- Ensemble muni d'un opérateur associatif $\langle E, op \rangle$

⇒ Loi de composition stable : $\forall x \in E, \forall y \in E \Rightarrow x op y \in E$

- Exemples

- $\langle \mathbb{N}, + \rangle$
- $\langle \text{ensemble}, \cup \rangle$
- $\langle \text{pile}, \text{poser} \rangle$

⇒ Élément neutre dans E

⇒ La structure $\langle L^*, . \rangle$ est un monoïde libre

Problème

- Langage des montants monétaires
 - Alphabet des chiffres
 - Combinaison de chiffres pour un nombre entier
 - Symbole pour les décimales et langage
 - Symboles des unités monétaires
- ⇒ Langage des montants

Plan

1. Symboles, alphabet et mots
2. Langages générés
3. Expressions régulières

Formalisations de langages

- Décrire un langage
 - **Définitoire** : langage naturel
 - Exemples sur $\Sigma = \{a, b\}$
 - « N'importe quel mot commence par ba »
 - « Dans un mot, tout a est suivi d'au moins un b »
 - « Dans un mot, il y a autant de a que de b »
 - **Algèbre** : concaténation, union, fermeture (...) de langages
 - **Expressions régulières** : expressions concises (programme)
 - **Grammaires** : formalisation générative (récursivité)

⇒ Formalisations plus/moins **puissantes** et **implémentables**

⇒ Langages **réguliers** par **expressions régulières**
ou langages / expressions **rationnel(le)s**

⇒ Couramment, **regex**

Expressions régulières basiques

- ▶ **Quantificateurs**
 - **Aucun opérateur** ab : concaténation
 - **Étoile de Kleene** a^* : répétition (entre 0 et $+\infty$)
 - ▶ **Caractères spéciaux**
 - **Point** $.$: n'importe quel caractère
 - **Chapeau** \wedge : début de ligne
 - **Dollar** $\$$: fin de ligne
 - **Antislash** $b \backslash b$: frontière de mot
 - **Classes de caractères** (sauf répétition, un seul symbole)
 - $[aei]$: liste de caractères
 - $[0-9]$ ou $[[:num:]]$: chiffres
 - $[[:alpha:]]$: caractères alphabétiques (dont diacritiques)
 - $[[:lower:]]$, $[[:upper:]]$: minuscules, majuscules, etc.
 - $[\wedge xyz]$: négation de classe
- ⇒ Combinaisons possibles : $[a-zA-Z]$

⇒ Utilisation : **grep**, **sed**, **awk**

Expressions régulières étendues

- ▶ **Parenthèses** (expr) : groupement (et capture)
- ⇒ Concaténation n'est pas prioritaire (parenthèses nécessaires)
- ▶ **Quantificateurs**
 - **Plus** (expr)+ : au moins une fois
 - **Point d'interrogation** (expr)? : au plus une fois
 - **Accolades** (expr){m,n} : itération entre m et n fois
- ▶ **Opérateur**
 - **Barre verticale** (expr1)|(expr2) : union (ou, disjonction)
- ⇒ Plus standard : egrep, perl, python (...)

Recherche d'occurrences

- ▶ **Stratégies** courantes de recherche d'occurrences
 - De gauche à droite dans la chaîne
 - Comportement *greedy* (gourmand) par défaut
 - ⇒ Sinon opérateur ? après le quantificateur
 - Pas de chevauchements
- ▶ Particularités des outils courants
 - Attention aux prises en charge caractères non-ASCII
 - **grep** (*globally search a regular expression and print*)
 - Recherche d'occurrences dans des fichiers « texte plein »
 - Ligne par ligne (pas d'occurrences à cheval sur deux lignes)
 - **sed** (*stream editor*)
 - Effacement ou remplacement dans des fichiers
 - Ligne par ligne (pas d'occurrences à cheval sur deux lignes)
 - Utilisation des captures pour remplacer

Références en ligne

- ▶ Référence <http://www.regular-expressions.info/>
- ▶ Perl / Python
https://www.johndcook.com/blog/python_regex/
- ▶ Unicode <http://unicode.org/reports/tr18/>
- ▶ ... (beaucoup, beaucoup d'autres)

Exercices

- ▶ Dans le corpus `80jours.txt`
 - Sélectionner en utilisant `egrep --color "regexp" 80jours.txt`
 - n'importe quelle ligne
 - occurrences de « merci »
 - les déclinaisons de « heureux »
 - des dates (par ex. années ou jours et mois)
 - les nombres écrits en lettres romaines (chapitres)
 - n'importe quel mot
 - des noms propres
 - les balises d'entités nommées

Exercices (suite)

- ▶ Dans le corpus `80jours.txt`
 - En utilisant

```
sed "s/regexp/repl/g" 80jours.txt
```

où `reg` est une expression régulière et `exp` son remplacement
 - Supprimer les ponctuations
 - Remplacer les verbes à l'infinitif par `INF`
 - Effacer les balises d'entités (`<per>`, `</pers>`, `<loc>`)
 - En utilisant

```
grep -o "regexp" 80jours.txt | sort | uniq -c | sort -gr
```

 - Compter le nombre d'entités par type
 - Compter les occurrences d'entités