

# Théorie des langages rationnels : THLR

CM 1

Uli Fahrenberg

EPITA Rennes

S3 2024

# Aperçu

# Ouverture culturelle

C'est quoi ce cours ?

- fondements **algébriques** de l'informatique
- fondements de la **calculabilité**
- ouverture scientifique
- la qualité fondamentale d'un chercheur :

# Ouverture culturelle

C'est quoi ce cours ?

- fondements **algébriques** de l'informatique
- fondements de la **calculabilité**
- ouverture scientifique
- la qualité fondamentale d'un chercheur : **la curiosité**

# Langages, mots, expressions

Langages :

- de programmation
- naturelles
- en bio-informatique, *etc.*

# Langages, mots, expressions

Langages :

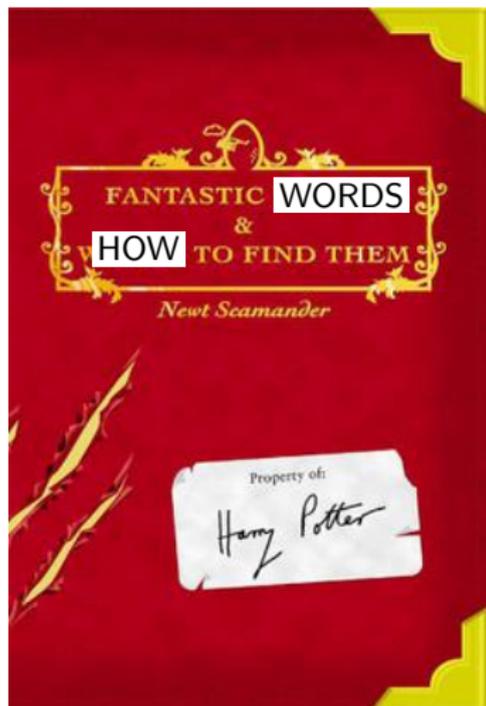
- de programmation
- naturelles
- en bio-informatique, *etc.*
- *qu'est-ce que* : **syntaxe**, **sémantique**

# Langages, mots, expressions

Langages :

- de programmation
- naturelles
- en bio-informatique, etc.
- *qu'est-ce que* : **syntaxe**, **sémantique**

Mots :



# Langages, mots, expressions

Langages :

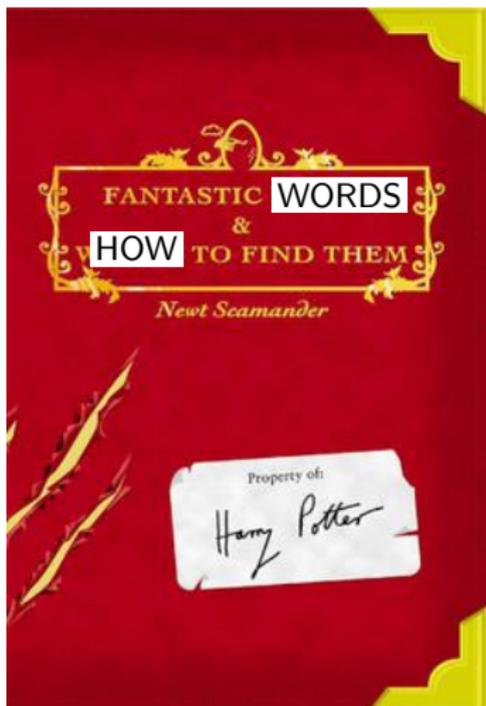
- de programmation
- naturelles
- en bio-informatique, *etc.*
- *qu'est-ce que* : **syntaxe**, **sémantique**

Mots :

- suite **finie** de *symboles*
- `while`, `my_var_336`,  
Schallplattenabspielgerät,  
`ACTAAGGT`

Expressions rationnelles :

- $[a-zA-Z][a-zA-Z0-9_]*$



# Un peu (!) de précision

Symbole :

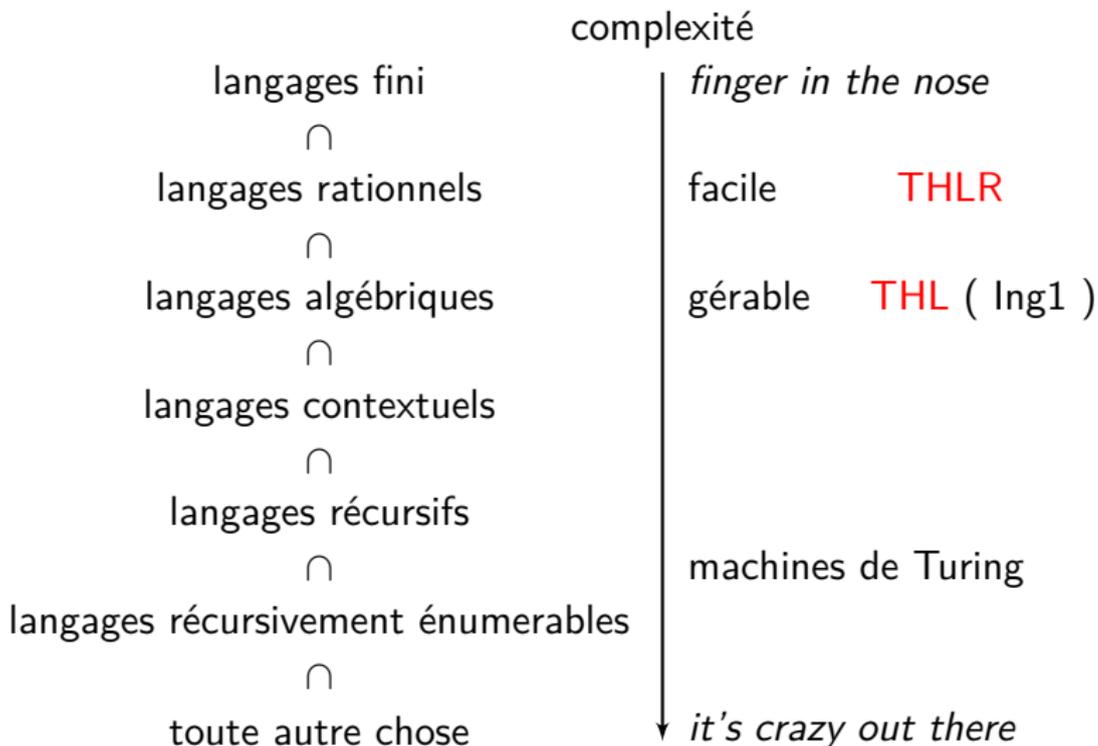
- notion axiomatique ( *on s'en fout de ce que c'est* )
- $\Sigma = \{a, b, c, \dots\}$

Mot :

- **suite finie** de symboles
- *a, abba, abracadabra, jenesaispasquoimaisilfautdubeurresurlepain*
- finie, mais sans limite fixe de longueur
- “Ich bin ein Berliner”, “for x in range(5)” ← pas souvent

Langage :

- **ensemble** de mots
- peut être fini ( même vide ! ), mais normalement **infini**

... et pourquoi *rationnel* ?

# Une démonstration

## Définition

Un langage  $L$  est **récursivement énumérable** s'il existe un algorithme qui énumère tous les mots de  $L$ .

Exemple : `x = 2`  
`while true:`  
    `if isprime(x): print(x)`  
    `x += 1`

# Une démonstration

## Définition

Un langage  $L$  est **rékursivement énumérable** s'il existe un algorithme qui énumère tous les mots de  $L$ .

## Théorème

*Il existe un langage qui n'est pas rékursivement énumérable.*

## Démonstration.

- 1 L'ensemble de tous les algorithmes est **dénombrable**. ( *Pourquoi ? Qu'est-ce que ?* )
- 2 Chaque algorithme n'énumère guère qu'un langage.
- 3 L'ensemble de langages n'est **pas dénombrable**. ( *Pourquoi ?* )

## ... et pourquoi ?

Des applications :

- le parsing
  - expressions rationnelles
  - `grep 'a.*io.*e.*e' thlr1.txt`
- la compilation
  - analyse lexicale
  - analyse syntaxique
- la bio-informatique
  - analyse de mutations
  - « Ève mitochondriale »
- la traduction automatique

# Pour en finir ( la première partie )

## Définition

Un algorithme  $A$  **décide** un langage donné  $L$  si, pour chaque mot  $w$  en entrée,  $A$  répond « OUI » si  $w \in L$  et « NON » si  $w \notin L$ .

## Exercice ( 5 mn )

- 1 Trouver un algorithme *simple* qui décide le langage de tous les mots qui **commencent par  $ab$**  :

$$L = \{ab, aba, abb, abaa, abab, abba, \dots\}$$

- 2 Trouver un algorithme *simple* qui décide le langage de tous les mots qui **se terminent par  $ab$**  :

$$L = \{ab, aab, bab, aaab, abab, baab, \dots\}$$

## Infos pratiques

# Les acquis d'apprentissages visés

À l'issue de ce cours, les étudiants doivent être capables de :

- Manipuler (préfixe, suffixe, facteur, distance, etc.) et exprimer des propriétés simples (décidabilité, régularité) sur les mots et les langages.
- Formuler un langage rationnel sous forme d'expression régulière.
- Interpréter et simplifier des expressions régulières.
- Convertir une expression régulière en un automate déterministe minimal.

# Les notes

Seront notés :

- les 4 meilleurs parmi 5 QCM / DM ( 15 % )
- les 4 meilleurs parmi 5 TP ( 20 % )
- l'examen mi-parcours ( 30 mn ; 25 % )
- l'examen final ( 60 mn ; 40 % )

# Programme d'aujourd'hui

- 1 Symboles, mots, langages
- 2 L'algèbre de langages
- 3 Opérations, relations et distances sur mots

# Programme du cours

- 1 Mots, langages
- 2 Langages rationnels, expressions rationnelles
- 3 Automates finis
- 4 Langages non-rationnels
- 5 Langages reconnaissables, minimisation

# Le poly

F.Yvon, A.Demaille, **Théorie des langages rationnels**

- cours  $\subsetneq$  shuffle(chapitres 1-4)
- aujourd'hui : chapitre 2, **moins** 2.3.2, 2.3.5, 2.4.4
- voir moodle://Supports

# Symboles, mots, langages

# Symboles, mots, langages : de la précision

Soit  $\Sigma$  un ensemble **fini**.

- on appelle  $\Sigma$  un **alphabet**
- et les éléments  $a, b, \dots \in \Sigma$  des **symboles**

On dénote  $\Sigma^*$  l'ensemble de toutes les **suites finies** d'éléments de  $\Sigma$ .

- donc  $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \dots = \bigcup_{n \geq 0} \Sigma^n$
- on appelle les éléments  $u, v, w, \dots \in \Sigma^*$  des **mots**
- on écrit des mots  $aabab$  ( par exemple ) au lieu de  $(a, a, b, a, b)$

Un **langage** est un sous-ensemble  $L \subseteq \Sigma^*$ .

# L'algèbre de mots

Il y a une opération binaire sur  $\Sigma^*$  :

# L'algèbre de mots

Il y a une opération binaire sur  $\Sigma^*$  :

## Définition

La **concaténation** de deux mots  $a_1 \dots a_n$  et  $b_1 \dots b_m$  est le mot  $a_1 \dots a_n b_1 \dots b_m$ .

- on utilise le symbole « . » si besoin ; sinon, rien

Voici les propriétés de la concaténation :

# L'algèbre de mots

Il y a une opération binaire sur  $\Sigma^*$  :

## Définition

La **concaténation** de deux mots  $a_1 \dots a_n$  et  $b_1 \dots b_m$  est le mot  $a_1 \dots a_n b_1 \dots b_m$ .

- on utilise le symbole « . » si besoin ; sinon, rien

Voici les propriétés de la concaténation :

## Théorème

L'opération « . » est **associative** et a **le mot vide comme élément neutre** de deux côtés.

- on utilise  $\varepsilon$  pour le mot vide
- donc  $u(vw) = (uv)w$ ,  $u.\varepsilon = u$  et  $\varepsilon.u = u$  pour tout  $u, v, w \in \Sigma^*$
- **pas commutative**

# L'algèbre de langages

# Opérations sur langages

## Opérations ensemblistes

# Opérations sur langages

## Opérations ensemblistes

$$L_1 \cup L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ ou } u \in L_2\}$$

$$L_1 \cap L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ et } u \in L_2\}$$

$$\bar{L} = \{u \in \Sigma^* \mid u \notin L\}$$

# Opérations sur langages

## Opérations ensemblistes

$$L_1 \cup L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ ou } u \in L_2\}$$

$$L_1 \cap L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ et } u \in L_2\}$$

$$\bar{L} = \{u \in \Sigma^* \mid u \notin L\}$$

## Concaténation

$$L_1 \cdot L_2 = \{u_1 u_2 \mid u_1 \in L_1, u_2 \in L_2\}$$

$$L^n = L \cdots L \quad (n \text{ copies de } L)$$

# Opérations sur langages

## Opérations ensemblistes

$$L_1 \cup L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ ou } u \in L_2\}$$

$$L_1 \cap L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ et } u \in L_2\}$$

$$\bar{L} = \{u \in \Sigma^* \mid u \notin L\}$$

## Concaténation

$$L_1 \cdot L_2 = \{u_1 u_2 \mid u_1 \in L_1, u_2 \in L_2\}$$

$$L^n = L \cdots L \quad (n \text{ copies de } L)$$

## Étoile de Kleene

$$L^* = L^0 \cup L^1 \cup L^2 \cup \dots = \bigcup_{n \geq 0} L^n$$

# L'algèbre de langages

## Théorème

L'opération « . » sur langages est **associative** et a le langage  $\{\varepsilon\}$  **comme élément neutre** de deux côtés.

- donc  $L_1(L_2L_3) = (L_1L_2)L_3$ ,  $L.\{\varepsilon\} = L$  et  $\{\varepsilon\}.L = L$
- **pas commutative**
- aussi,  $L.\emptyset = \emptyset$  et  $\emptyset.L = \emptyset$

## Théorème

$\Sigma^* = \Sigma^*$ .

- ( ce n'est pas une tautologie )
- aussi,  $\emptyset^* = \{\varepsilon\}$ , en fait  $\varepsilon \in L^*$  pour chaque  $L$

# 5 minutes de réflexion

Vrai ou faux ?

①  $\{ab\} \cup \{ba\} = \{abba\}$

②  $\{a\}^n = \{a^n\}$

③  $\{a\}^* = \{a^n \mid n \geq 0\}$

④  $\{a, b\}^n = \{a^n, b^n\}$

⑤  $\{a, b\}^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$

⑥  $(L_1 \cup L_2)^2 = L_1^2 \cup L_1L_2 \cup L_2L_1 \cup L_2^2$

⑦  $L_1 \cdot (L_2 \cup L_3) = L_1L_2 \cup L_1L_3$

## 5 minutes de réflexion

Vrai ou faux ?

- ①  $\{ab\} \cup \{ba\} = \{abba\}$  ✗
- ②  $\{a\}^n = \{a^n\}$  ✓
- ③  $\{a\}^* = \{a^n \mid n \geq 0\}$  ✓
- ④  $\{a, b\}^n = \{a^n, b^n\}$  ✗
- ⑤  $\{a, b\}^3 = \{aaa, aab, aba, abb, baa, bab, bba, bbb\}$  ✓
- ⑥  $(L_1 \cup L_2)^2 = L_1^2 \cup L_1L_2 \cup L_2L_1 \cup L_2^2$  ✓
- ⑦  $L_1 \cdot (L_2 \cup L_3) = L_1L_2 \cup L_1L_3$  ✓

# Opérations sur mots

# Longueur d'un mot

## Définition

La **longueur**  $|u|$  d'un mot  $u \in \Sigma^*$  correspond au nombre de symboles de  $u$ .

- ( alors  $|ababa| = 5$  )
- $|\varepsilon| = 0$  et  $|uv| = |u| + |v|$
- aussi,  $|u| = 0$  ssi  $u = \varepsilon$
- et  $|u| = 1$  ssi  $u \in \Sigma$

## Notation

On dénote  $u^n$  la concaténation de  $n$  copies de  $u \in \Sigma^*$ .

- donc  $(abc)^3 = abcabcabc$
- définition récursive :  $u^0 = \varepsilon$  et  $u^{n+1} = u u^n$
- pour la longueur,  $|u^n| = n|u|$

# Préfixe, suffixe, facteur

## Définition

Soit  $u, v \in \Sigma^*$ , alors  $u$  est un **préfixe** de  $v$  ssi il existe  $w \in \Sigma^*$  tel que  $uw = v$ .

- des préfixes de *tomate* :

$\{ t, to, tom, toma, tomat, tomate \}$

# Préfixe, suffixe, facteur

## Définition

Soit  $u, v \in \Sigma^*$ , alors  $u$  est un **préfixe** de  $v$  ssi il existe  $w \in \Sigma^*$  tel que  $uw = v$ .

- des préfixes de *tomate* :

$\{\varepsilon, t, to, tom, toma, tomat, tomate\}$

# Préfixe, suffixe, facteur

## Définition

Soit  $u, v \in \Sigma^*$ , alors  $u$  est un **préfixe** de  $v$  ssi il existe  $w \in \Sigma^*$  tel que  $uw = v$ .

- des préfixes de *tomate* :

$\{\varepsilon, t, to, tom, toma, tomat, tomate\}$

## Définition

Soit  $u, v \in \Sigma^*$ , alors

- $u$  est un **suffixe** de  $v$  ssi  $\exists w \in \Sigma^* : wu = v$  ;
- $u$  est un **facteur** de  $v$  ssi  $\exists w_1, w_2 \in \Sigma^* : w_1 u w_2 = v$ .

## Préfixe, suffixe, facteur, 2.

Pour un langage  $L \subseteq \Sigma^*$  on note le **langage de préfixes** de  $L$  par

$$\text{Pref}(L) = \{u \in \Sigma^* \mid \exists v \in L : u \text{ préfixe de } v\}$$

- donc  $\text{Pref}(\{\text{tomate}\}) = \{\varepsilon, t, to, tom, toma, tomat, tomate\}$
- même chose pour  $\text{Suff}(L)$  et  $\text{Fact}(L)$

**Vrai ou faux ?** ( 5 mn )

- 1  $\text{Fact}(L) = \text{Pref}(L) \cup \text{Suff}(L)$
- 2  $\text{Pref}(\text{Pref}(L)) = \text{Pref}(L)$
- 3  $\text{Pref}(\text{Fact}(L)) = \text{Pref}(L)$
- 4  $\text{Pref}(\text{Fact}(L)) = \text{Fact}(L)$
- 5  $\text{Pref}(\text{Suff}(L)) = \text{Suff}(\text{Pref}(L)) = \text{Fact}(L)$

## Préfixe, suffixe, facteur, 2.

Pour un langage  $L \subseteq \Sigma^*$  on note le **langage de préfixes** de  $L$  par

$$\text{Pref}(L) = \{u \in \Sigma^* \mid \exists v \in L : u \text{ préfixe de } v\}$$

- donc  $\text{Pref}(\{tomate\}) = \{\varepsilon, t, to, tom, toma, tomat, tomate\}$
- même chose pour  $\text{Suff}(L)$  et  $\text{Fact}(L)$

**Vrai ou faux ?** ( 5 mn )

- 1  $\text{Fact}(L) = \text{Pref}(L) \cup \text{Suff}(L)$  ✗
- 2  $\text{Pref}(\text{Pref}(L)) = \text{Pref}(L)$  ✓
- 3  $\text{Pref}(\text{Fact}(L)) = \text{Pref}(L)$  ✗
- 4  $\text{Pref}(\text{Fact}(L)) = \text{Fact}(L)$  ✓
- 5  $\text{Pref}(\text{Suff}(L)) = \text{Suff}(\text{Pref}(L)) = \text{Fact}(L)$  ✓

## Relations d'ordre sur mots

# L'ordre de préfixe

Écrivons  $u \leq_p v$  si  $u$  est un préfixe de  $v$

- donc  $\text{Pref}(L) = \{u \in \Sigma^* \mid \exists v \in L : u \leq_p v\}$

La relation  $\leq_p$  sur mots est

- réflexive

$$u \leq_p u$$

- transitive

$$u \leq_p v \text{ et } v \leq_p w \Rightarrow u \leq_p w$$

- antisymétrique

$$u \leq_p v \text{ et } v \leq_p u \Rightarrow u = v$$

$\Rightarrow$  une relation d'ordre partiel

- $tom \leq_p tomate$ , mais  $tomate \not\leq_p patate$  et  $patate \not\leq_p tomate$

# L'ordre lexicographique

L'ordre lexicographique  $\hat{=}$  l'ordre du **dictionnaire** :

- $tom \leq_\ell tomate \leq_\ell tupac \leq_\ell ukulele$
- un ordre **total**

## Définition

Soit  $\leq$  un ordre total sur  $\Sigma$  et  $u, v \in \Sigma^*$ , alors on écrit  $u \leq_\ell v$  si

- $u \leq_p v$  ou
-

# L'ordre lexicographique

L'ordre lexicographique  $\hat{=}$  l'ordre du **dictionnaire** :

- $tom \leq_\ell tomate \leq_\ell tupac \leq_\ell ukulele$
- un ordre **total**

## Définition

Soit  $\leq$  un ordre total sur  $\Sigma$  et  $u, v \in \Sigma^*$ , alors on écrit  $u \leq_\ell v$  si

- $u \leq_p v$  ou
- $\exists a, b \in \Sigma, w, u', v' \in \Sigma^*$  t.q.  $u = wau'$ ,  $v = wbv'$  et  $a \leq b$

# L'ordre lexicographique

L'ordre lexicographique  $\hat{=}$  l'ordre du **dictionnaire** :

- $tom \leq_\ell tomate \leq_\ell tupac \leq_\ell ukulele$
- un ordre **total**

## Définition

Soit  $\leq$  un ordre total sur  $\Sigma$  et  $u, v \in \Sigma^*$ , alors on écrit  $u \leq_\ell v$  si

- $u \leq_p v$  ou
- $\exists a, b \in \Sigma, w, u', v' \in \Sigma^*$  t.q.  $u = wau', v = wbv'$  et  $a \leq b$

Problèmes théoriques : l'ordre  $\leq_\ell$

- n'est **pas compatible** avec la concaténation :  $a \leq_\ell ab$  mais  $a.c \not\geq_\ell ab.c$
- n'est **pas noëthérien** / **pas bien fondé** : il existe des suites infinies strictement décroissantes
  - par exemple,  $b \geq_\ell ab \geq_\ell aab \geq_\ell aaab \geq_\ell \dots$

# L'ordre radiciel

L'ordre **radiciel**, ou **militaire** :

## Définition

Soit  $\leq$  un ordre total sur  $\Sigma$  et  $u, v \in \Sigma^*$ , alors on écrit  $u \leq_r v$  si

- $|u| < |v|$  ou
  - $|u| = |v|$  et  $u \leq_\ell v$
- 
- compatible avec la concaténation
  
  - noëthérien
  
  - mais pas compatible avec l'ordre lexicographique

# L'ordre radiciel

L'ordre **radiciel**, ou **militaire** :

## Définition

Soit  $\leq$  un ordre total sur  $\Sigma$  et  $u, v \in \Sigma^*$ , alors on écrit  $u \leq_r v$  si

- $|u| < |v|$  ou
- $|u| = |v|$  et  $u \leq_\ell v$

- compatible avec la concaténation

$$u \leq_r v \Rightarrow uw \leq_r vw \text{ et } wu \leq_r wv$$

- noëthérien

$$u_1 \geq_r u_2 \geq_r u_3 \geq_r \dots \Rightarrow \exists m : \forall n \geq m : u_n = u_m$$

- mais pas compatible avec l'ordre lexicographique

$$\text{par exemple, } b \leq_r ab \text{ mais } ab \not\leq_\ell b$$

# Distances entre mots

# La distance préfixe

Notons  $\text{plpc}(u, v)$  le plus long préfixe commun entre mots  $u$  et  $v$

# La distance préfixe

Notons  $\text{plpc}(u, v)$  le plus long préfixe commun entre mots  $u$  et  $v$

$$\begin{aligned}\text{plpc}(u, v) &= \max_{\leq_p} \{w \in \Sigma^* \mid w \leq_p u \text{ et } w \leq_p v\} \\ &= \max_{\leq_p} (\text{Pref}(\{u\}) \cap \text{Pref}(\{v\}))\end{aligned}$$

# La distance préfixe

Notons  $\text{plpc}(u, v)$  le plus long préfixe commun entre mots  $u$  et  $v$

$$\begin{aligned}\text{plpc}(u, v) &= \max_{\leq_p} \{w \in \Sigma^* \mid w \leq_p u \text{ et } w \leq_p v\} \\ &= \max_{\leq_p} (\text{Pref}(\{u\}) \cap \text{Pref}(\{v\}))\end{aligned}$$

## Définition

La **distance préfixe** entre mots  $u, v \in \Sigma^*$  est

$$d_p(u, v) = |uv| - 2|\text{plpc}(u, v)|.$$

Propriétés :

- **positivité**
  - **séparation**
  - **symétrie**
  - **inégalité triangulaire**
- ⇒  $d_p$  est, en fait, une **distance**

# La distance préfixe

Notons  $\text{plpc}(u, v)$  le plus long préfixe commun entre mots  $u$  et  $v$

$$\begin{aligned} \text{plpc}(u, v) &= \max_{\leq_p} \{w \in \Sigma^* \mid w \leq_p u \text{ et } w \leq_p v\} \\ &= \max_{\leq_p} (\text{Pref}(\{u\}) \cap \text{Pref}(\{v\})) \end{aligned}$$

## Définition

La **distance préfixe** entre mots  $u, v \in \Sigma^*$  est

$$d_p(u, v) = |uv| - 2|\text{plpc}(u, v)|.$$

Propriétés :

- **positivité**
  - **séparation**
  - **symétrie**
  - **inégalité triangulaire**
- $\Rightarrow d_p$  est, en fait, une **distance**

$$d_p(u, v) \geq 0$$

$$d_p(u, v) = 0 \iff u = v$$

$$d_p(u, v) = d_p(v, u)$$

$$d_p(u, w) \leq d_p(u, v) + d_p(v, w)$$

# La distance d'édition

Des opérations élémentaires sur mots, pour  $u, v \in \Sigma^*$  et  $a \in \Sigma$  :

- insertion :  $uv \xrightarrow{+a} uav$
- suppression :  $uav \xrightarrow{-a} uv$

## Définition

La **distance d'édition** (ou de *Levenshtein*) entre  $u, v \in \Sigma^*$  est la longueur minimale d'une séquence d'opérations élémentaires entre  $u$  et  $v$ .

Exemple :  $tomate \xrightarrow{-e} tomat \xrightarrow{-o} tmat \xrightarrow{+u} tumat \xrightarrow{-m} tuat \xrightarrow{+p}$   
 $tupat \xrightarrow{-t} tupa \xrightarrow{+c} tupac \Rightarrow \text{distance} \leq 7$

## Applications :

- GNU diff
- correcteurs orthographique
- reconnaissance de texte
- analyse de mutations (!)

The image features a central graphic consisting of several concentric circles. The innermost circle is a solid dark blue. Surrounding it are several rings of varying shades of red, from a bright, almost white-red to a deep, dark red. The outermost ring is black. Overlaid on this graphic is the text "That's all Folks!" written in a white, elegant cursive script. The text is positioned diagonally across the center of the circles, starting from the left side and ending on the right side. The overall composition is balanced and visually striking due to the high contrast between the reds, black, and white.

*That's all Folks!*