Théorie des langages : THL

CM 1

Uli Fahrenberg

EPITA Rennes

S5 2023

Aperçu

Langages, mots, expressions

Langages :

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

Langages, mots, expressions

Langages :

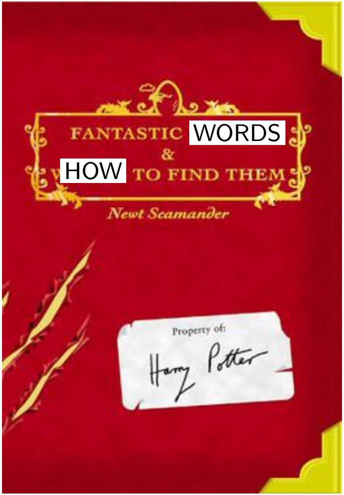
- de programmation
- langues naturelles
- en bio-informatique, *etc.*
- *qu'est-ce que :* syntaxe, sémantique

Langages, mots, expressions

Langages :

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

Mots :



Langages, mots, expressions

Langages :

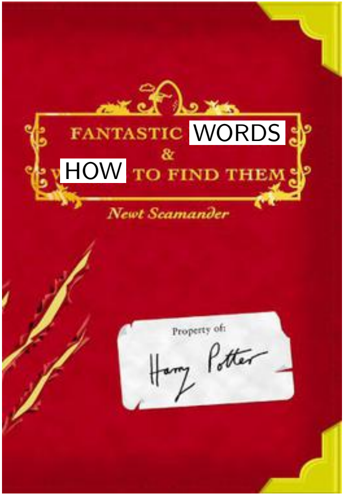
- de programmation
- langues naturelles
- en bio-informatique, *etc.*
- *qu'est-ce que :* syntaxe, sémantique

Mots :

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

Expressions rationnelles :

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



Mkrtchian

Մհեր Մուշեղի Մկրտչյան

Мгер Мушегович Мкртчян

Mher Mouchérovitch Mkrtchian

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)`" (!)

Langage :

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

Classification

		complexité
langages fini		*finger in the nose*
$\cap$		
langages rationnels		facile
$\cap$		THL
langages algébriques		gérable
$\cap$		
langages contextuels		
$\cap$		
langages récursifs		
$\cap$		machines de Turing
langages récursivement énumerables		
$\cap$		
toute autre chose	$\downarrow$	*it's crazy out there*

Une démonstration

Définition

Un langage L est récursivement énumerable s'il existe un algorithme qui énumère tout les mots de L.

Exemple :

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

Une démonstration

Définition

Un langage L est récursivement énumerable s'il existe un algorithme qui énumère tout les mots de L.

Théorème

Il existe un langage qui n'est pas récursivement énumerable.

Démonstration.

1. L'ensemble de tous 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 parsage
  - 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

Le cours

1. Langages rationnels
2. Automates finis

3. Langages algébriques, grammaires hors-contexte
4. Automates à pile
5. Parsage LL

6. Parsage LR

- et après, le Tiger project !

Le cours

1. Langages rationnels
2. Automates finis
   - TP `flex`
3. Langages algébriques, grammaires hors-contexte
4. Automates à pile
5. Parsage LL
   - TP LL
6. Parsage LR
   - TP `bison`
   - TP `flex` et `bison`

- et après, le Tiger project !

Les notes

1. Langages rationnels
2. Automates finis
   TP `flex` — QCM
3. Langages algébriques, grammaires hors-contexte
4. Automates à pile
5. Parsage LL
   TP LL — QCM
6. Parsage LR
   TP `bison` — QCM
   TP `flex` et `bison`

- et après, le Tiger project !

Le prof

Uli Fahrenberg
https://ulifahrenberg.github.io/
uli.fahrenberg@epita.fr

Le poly

F.Yvon, A.Demaille, Théorie des langages

- cours $\subsetneq$ shuffle(chapitres 1-8)
- aujourd'hui :
  - chapitre 2, moins 2.3.2-5 et 2.4.4
  - chapitre 3, moins 3.1.3

- Moodle
- `https://ulifahrenberg.github.io/thl/`

Programme d'aujourd'hui

1. Symboles, mots, langages
2. L'algèbre de langages
3. Langages rationnels
4. Expressions rationnelles

Symboles, mots, langages

Symboles, mots, langages : de la précision

Soit Σ un ensemble fini.

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

On dénote Σ^* l'ensemble de tous les suites finies d'éléments de Σ.

- donc $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots = \bigcup_{n \geq 0} \Sigma^n$
- on appelle les éléments $u, v, w, \ldots \in \Sigma^*$ 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 Σ^* :

L'algèbre de mots

Il y a une opération binaire sur Σ^* :

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 Σ^* :

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 ε 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

Opérations sur langages

Opérations ensemblistes

Opérations sur langages

Opérations ensemblistes

$$L_1 \cup L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ ou } u \in L_2\}$$
$$L_1 \cap L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ et } u \in L_2\}$$
$$\overline{L} = \{u \in \Sigma^* \mid u \notin L\}$$

Opérations sur langages

Opérations ensemblistes

$$L_1 \cup L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ ou } u \in L_2\}$$
$$L_1 \cap L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ et } u \in L_2\}$$
$$\overline{L} = \{u \in \Sigma^* \mid u \notin L\}$$

Concaténation

$$L_1.L_2 = \{u_1\, u_2 \mid u_1 \in L_1, u_2 \in L_2\}$$
$$L^n = L \cdots L \quad (n \text{ copies de } L)$$

Opérations sur langages

Opérations ensemblistes

$$L_1 \cup L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ ou } u \in L_2\}$$
$$L_1 \cap L_2 = \{u \in \Sigma^* \mid u \in L_1 \text{ et } u \in L_2\}$$
$$\overline{L} = \{u \in \Sigma^* \mid u \notin L\}$$

Concaténation

$$L_1.L_2 = \{u_1\, u_2 \mid u_1 \in L_1, u_2 \in L_2\}$$
$$L^n = L \cdots L \quad (n \text{ copies de } L)$$

Étoile de Kleene

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

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. $(L_1 \cup L_2)^2 = L_1^2 \cup L_1 L_2 \cup L_2 L_1 \cup L_2^2$
7. $L_1.(L_2 \cup L_3) = L_1 L_2 \cup L_1 L_3$

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. $(L_1 \cup L_2)^2 = L_1^2 \cup L_1 L_2 \cup L_2 L_1 \cup L_2^2$ ✓
7. $L_1.(L_2 \cup L_3) = L_1 L_2 \cup L_1 L_3$ ✓

Longueur d'un mot

Définition

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

- ($|ababa| = 5$, pas 2 !)
- donc $|\varepsilon| = 0$ 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 $u^{n+1} = u\, u^n$
- aussi, $|u^n| = n\,|u|$

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* :

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

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 : suite

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

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

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

Vrai ou faux ? (5 mn)

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

Préfixe, suffixe, facteur : suite

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

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

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

Vrai ou faux ? (5 mn)

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

Exercice

Redéfinissez chacun des langages suivants, sur alphabet $\Sigma = \{a, b\}$, en n'utilisant que des ensembles finis et des opérations $\cup$, . et * :

1. $\mathrm{Pref}(\{a\}\{b\}^*)$
2. $\mathrm{Suff}(\{a\}\{b\}^*)$
3. $\{a\}\{b\}^*\{a\}^* \cap \{a\}^*\{b\}^*\{a\}$
4. $\overline{\{a\}^*}$

Exercice

Redéfinissez chacun des langages suivants, sur alphabet $\Sigma = \{a, b\}$, en n'utilisant que des ensembles finis et des opérations $\cup$, . et * :

1. $\mathrm{Pref}(\{a\}\{b\}^*) = \{\varepsilon, a\} \cup \{a\}\{b\}^*$
2. $\mathrm{Suff}(\{a\}\{b\}^*) = \{a\}\{b\}^* \cup \{b\}^*$
3. $\{a\}\{b\}^*\{a\}^* \cap \{a\}^*\{b\}^*\{a\} = \{a\}\{b\}^*\{a\} \cup \{a\}\{a\}^*$
4. $\overline{\{a\}^*} = \{a, b\}^*\{b\}\{a, b\}^*$

Langages rationnels

5 minutes de réflexion

Vrai ou faux ?

1. $\{a\}^n = \{a^n\}$
2. $\{a, b\}^n = \{a^n, b^n\}$
3. L^* est un ensemble infini pour tout $L \subseteq \Sigma^*$
4. pour tout $L \subseteq \Sigma^*$ et $n \in \mathbb{N}$, l'ensemble $\{u \in L \mid |u| \leq n\}$ est un ensemble fini
5. $\{a, b\}^* = \{a\}^*\{b\}^*$
6. $\{a, b\}^* = \big(\{a\}^*\{b\}\big)^*\{a\}^*$
7. $(L_1 \cup L_2)^* = \big(L_1^* L_2\big)^* L_1^*$

5 minutes de réflexion

Vrai ou faux ?

1. $\{a\}^n = \{a^n\}$ ✓
2. $\{a, b\}^n = \{a^n, b^n\}$ ✗
3. L^* est un ensemble infini pour tout $L \subseteq \Sigma^*$ ✗
4. pour tout $L \subseteq \Sigma^*$ et $n \in \mathbb{N}$, l'ensemble $\{u \in L \mid |u| \leq n\}$ est un ensemble fini ✓
5. $\{a, b\}^* = \{a\}^*\{b\}^*$ ✗
6. $\{a, b\}^* = \big(\{a\}^*\{b\}\big)^*\{a\}^*$ ✓
7. $(L_1 \cup L_2)^* = \big(L_1^* L_2\big)^* L_1^*$ ✓

Opérations rationnelles

Soit Σ un alphabet, on travaille avec des langages dans $\mathcal{P}(\Sigma^*)$.

On a vu dans l'exo que les opérations $\cup$, . et * sont bien spéciales.

- $\text{Pref}(\{a\}\{b\}^*) = \{\varepsilon, a\} \cup \{a\}\{b\}^*$
- $\text{Suff}(\{a\}\{b\}^*) = \{a\}\{b\}^* \cup \{b\}^*$
- etc.

Définition

Les opérations rationnelles dans $\mathcal{P}(\Sigma^*)$ sont $\cup$, . et *.

- donc union, concaténation et étoile de Kleene

Théorème (pour plus tard) : Toutes les autres opérations sont exprimables par $\cup$, . et *.

Langages rationnels

Définition (3.1)

Les langages rationnels sur Σ sont définis inductivement comme suite :

1. $\emptyset$ et $\{\varepsilon\}$ sont des langages rationnels
2. pour tout $a \in \Sigma$, $\{a\}$ est un langage rationnel
3. si L_1 et L_2 sont des langages rationnels, alors $L_1 \cup L_2$, $L_1 . L_2$ et L_1^* le sont également

- $\{\varepsilon\} = \emptyset^* \Rightarrow$ on peut enlever $\{\varepsilon\}$ de la définition

Lemme

L est rationnel si et seulement si

- $L = \emptyset$ *ou* $L = \{a\}$ *pour un* $a \in \Sigma$ *ou*
- $L = L_1 \cup L_2$, $L = L_1 L_2$ *ou* $L = L_1^*$ *pour* L_1 *et* L_2 *rationnels.*

(En quoi ce lemme est-il différent de la définition ?)

Rationalité

Théorème

Si L_1 et L_2 sont des langages rationnels, alors $L_1 \cap L_2$, $\overline{L}_1$, $\mathrm{Pref}(L_1)$, $\mathrm{Suff}(L_1)$ et $\mathrm{Fact}(L_1)$ le sont aussi.

- pour la démonstration faut attendre un peu

5 minutes de réflexion

Rationnel ou pas rationnel, sur alphabet $\Sigma = \{a, b, c\}$?

1. $\{a, b, abcba\}$
2. $\{a^n \mid n \geq 0\}$
3. $\{w \in \Sigma^* \mid w \text{ contient au moins trois } a\}$
4. $\{w \in \Sigma^* \mid |w| \geq 5\}$
5. $\{a^{2n} \mid n \geq 0\}$
6. $\{a^{n^2} \mid n \geq 0\}$
7. $\{a^m b^n \mid m, n \geq 0\}$
8. $\{a^n b^n \mid n \geq 0\}$

5 minutes de réflexion

Rationnel ou pas rationnel, sur alphabet $\Sigma = \{a, b, c\}$?

1. $\{a, b, abcba\}$ ✓
2. $\{a^n \mid n \geq 0\}$ ✓
3. $\{w \in \Sigma^* \mid w \text{ contient au moins trois } a\}$ ✓
4. $\{w \in \Sigma^* \mid |w| \geq 5\}$ ✓
5. $\{a^{2n} \mid n \geq 0\}$ ✓
6. $\{a^{n^2} \mid n \geq 0\}$ ✗
7. $\{a^m b^n \mid m, n \geq 0\}$ ✓
8. $\{a^n b^n \mid n \geq 0\}$ ✗

Expressions rationnelles

Expressions rationnelles

Une notation pratique pour des langages rationnels :

Définition (3.2)

Les expressions rationnelles sur Σ sont définis inductivement comme suite :

1. $\emptyset$ et ε sont des expressions rationnelles
2. pour tout $a \in \Sigma$, a est une expression rationnelle
3. si e_1 et e_2 sont des expressions rationnelles, alors $e_1 + e_2$, $e_1 . e_2$ et e_1^* le sont également

Expressions rationnelles

Une notation pratique pour des langages rationnels :

Définition (3.1, *recall*)

Les langages rationnels sur Σ sont définis inductivement comme suite :

1. $\emptyset$ et $\{\varepsilon\}$ sont des langages rationnels
2. pour tout $a \in \Sigma$, $\{a\}$ est un langage rationnel
3. si L_1 et L_2 sont des langages rationnels, alors $L_1 \cup L_2$, $L_1 . L_2$ et L_1^* le sont également

- presque la même chose ! mais
- 3.1 introduit une classe de sous-ensembles de Σ^*,
- 3.2 définit des expressions syntaxiques

Expressions rationnelles

Une notation pratique pour des langages rationnels :

Définition (3.2)

Les expressions rationnelles sur Σ sont définis inductivement comme suite :

1. $\emptyset$ et ε sont des expressions rationnelles
2. pour tout $a \in \Sigma$, a est une expression rationnelle
3. si e_1 et e_2 sont des expressions rationnelles, alors $e_1 + e_2$, $e_1 . e_2$ et e_1^* le sont également

- presque la même chose ! mais
- 3.1 introduit une classe de sous-ensembles de Σ^*,
- 3.2 définit des expressions syntaxiques

On va relier les deux en donnant une sémantique aux expressions rationnelles.

Sémantique

Définition

Le langage dénoté par une expression rationnelle e sur Σ est $L(e) \subseteq \Sigma^*$ définit inductivement comme suite :

1. $L(\varnothing) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$
2. $L(a) = \{a\}$ pour tout $a \in \Sigma$
3. $L(e_1 + e_2) = L(e_1) \cup L(e_2)$, $L(e_1 . e_2) = L(e_1) . L(e_2)$, $L(e^*) = (L(e))^*$

Théorème

$L \subseteq \Sigma^*$ est rationnel ssi il existe une expression rationnelle e telle que $L = L(e)$.

Démonstration.

Par induction structurelle (sur tableau).

La démonstration (sur tableau)

Les langages rationnels sur Σ :

1. $\emptyset$ et $\{\varepsilon\}$ sont des langages rationnels
2. pour tout $a \in \Sigma$, $\{a\}$ est un langage rationnel
3. L_1 et L_2 langages rationnels $\Rightarrow L_1 \cup L_2$, $L_1.L_2$ et L_1^* aussi

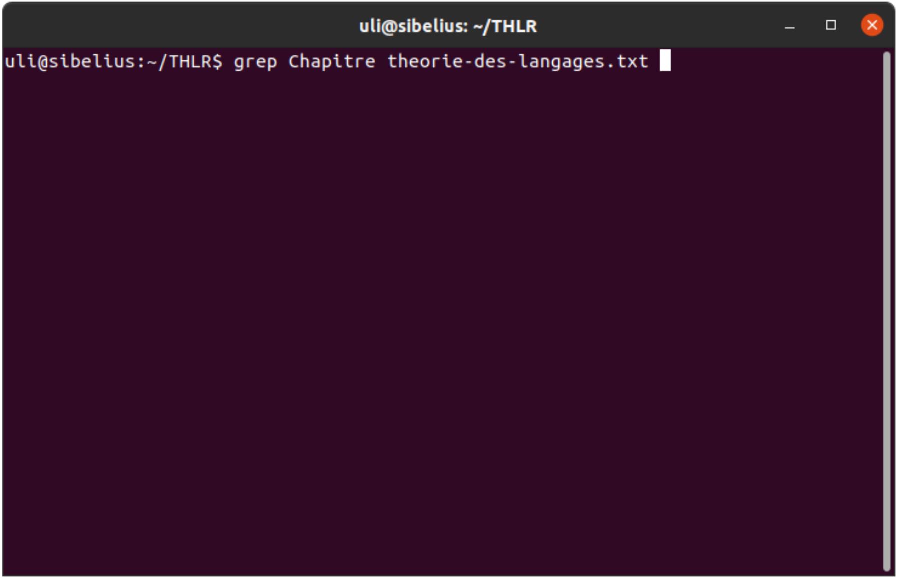
Les expressions rationnelles sur Σ :

1. $\varnothing$ et ε sont des expressions rationnelles
2. pour tout $a \in \Sigma$, a est une expression rationnelle
3. e_1 et e_2 expressions rationnelles $\Rightarrow e_1 + e_2$, $e_1.e_2$ et e_1^* aussi

Le langage dénoté par e :

1. $L(\varnothing) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$
2. $L(a) = \{a\}$ pour tout $a \in \Sigma$
3. $L(e_1 + e_2) = L(e_1) \cup L(e_2)$, $L(e_1.e_2) = L(e_1).L(e_2)$, $L(e^*) = (L(e))^*$

Exemples


uli@sibelius: ~/THLR
uli@sibelius:~/THLR$ grep Chapitre theorie-des-langages.txt

Exemples

uli@sibelius: ~/THLR

```
uli@sibelius:~/THLR$ grep Chapitre theorie-des-langages.txt
Chapitre 1
Chapitre 2
Chapitre 3
Chapitre 4
Chapitre 5
Chapitre 6
Chapitre 7
Chapitre 8
Chapitre 9
Chapitre 10
Chapitre rédigé par Pierre Senellart.
Chapitre 11
Chapitre rédigé par Pierre Senellart.
Chapitre 12
Chapitre 13
Chapitre 14
Chapitre 15
Chapitre 16
Chapitre 17
Chapitre 18
uli@sibelius:~/THLR$
```

Exemples

uli@sibelius: ~/THLR

```
uli@sibelius:~/THLR$ grep 'Chapitre [0-9]\+' theorie-des-langages.txt
```

Exemples

uli@sibelius: ~/THLR

```
uli@sibelius:~/THLR$ grep 'Chapitre [0-9]\+' theorie-des-langages.txt
Chapitre 1
Chapitre 2
Chapitre 3
Chapitre 4
Chapitre 5
Chapitre 6
Chapitre 7
Chapitre 8
Chapitre 9
Chapitre 10
Chapitre 11
Chapitre 12
Chapitre 13
Chapitre 14
Chapitre 15
Chapitre 16
Chapitre 17
Chapitre 18
uli@sibelius:~/THLR$
```

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

1. Soit e une expression rationnelle telle que $L = L(e)$.

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

1. Soit e une expression rationnelle telle que $L = L(e)$.
2. Nous construirons une expression rationnelle $\mathrm{pref}(e)$ telle que $L(\mathrm{pref}(e)) = \mathrm{Pref}(L)$.

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

1. Soit e une expression rationnelle telle que $L = L(e)$.
2. Nous construirons une expression rationnelle $\mathrm{pref}(e)$ telle que $L(\mathrm{pref}(e)) = \mathrm{Pref}(L)$.
3. … par induction structurelle :

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

1. Soit e une expression rationnelle telle que $L = L(e)$.
2. Nous construirons une expression rationnelle $\mathrm{pref}(e)$ telle que $L(\mathrm{pref}(e)) = \mathrm{Pref}(L)$.
3. … par induction structurelle :
4. $\mathrm{pref}(\emptyset) =$, $\mathrm{pref}(\varepsilon) =$, $\mathrm{pref}(a) =$

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

1. Soit e une expression rationnelle telle que $L = L(e)$.
2. Nous construirons une expression rationnelle $\mathrm{pref}(e)$ telle que $L(\mathrm{pref}(e)) = \mathrm{Pref}(L)$.
3. … par induction structurelle :
4. $\mathrm{pref}(\emptyset) = \emptyset$, $\mathrm{pref}(\varepsilon) = \varepsilon$, $\mathrm{pref}(a) = a + \varepsilon$

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

1. Soit e une expression rationnelle telle que $L = L(e)$.
2. Nous construirons une expression rationnelle $\mathrm{pref}(e)$ telle que $L(\mathrm{pref}(e)) = \mathrm{Pref}(L)$.
3. … par induction structurelle :
4. $\mathrm{pref}(\varnothing) = \varnothing$, $\mathrm{pref}(\varepsilon) = \varepsilon$, $\mathrm{pref}(a) = a + \varepsilon$
5. $\mathrm{pref}(e_1 + e_2) =$

   $\mathrm{pref}(e_1 e_2) =$

   $\mathrm{pref}(e^*) =$

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

1. Soit e une expression rationnelle telle que $L = L(e)$.
2. Nous construirons une expression rationnelle $\mathrm{pref}(e)$ telle que $L(\mathrm{pref}(e)) = \mathrm{Pref}(L)$.
3. ... par induction structurelle :
4. $\mathrm{pref}(\emptyset) = \emptyset$, $\mathrm{pref}(\varepsilon) = \varepsilon$, $\mathrm{pref}(a) = a + \varepsilon$
5. $\mathrm{pref}(e_1 + e_2) = \mathrm{pref}(e_1) + \mathrm{pref}(e_2)$
   $\mathrm{pref}(e_1 e_2) =$
   $\mathrm{pref}(e^*) =$

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

1. Soit e une expression rationnelle telle que $L = L(e)$.
2. Nous construirons une expression rationnelle $\mathrm{pref}(e)$ telle que $L(\mathrm{pref}(e)) = \mathrm{Pref}(L)$.
3. ... par induction structurelle :
4. $\mathrm{pref}(\varnothing) = \varnothing$, $\mathrm{pref}(\varepsilon) = \varepsilon$, $\mathrm{pref}(a) = a + \varepsilon$
5. $$\begin{aligned} \mathrm{pref}(e_1 + e_2) &= \mathrm{pref}(e_1) + \mathrm{pref}(e_2) \\ \mathrm{pref}(e_1 e_2) &= \mathrm{pref}(e_1) + e_1\, \mathrm{pref}(e_2) \\ \mathrm{pref}(e^*) &= \end{aligned}$$

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

1. Soit e une expression rationnelle telle que $L = L(e)$.
2. Nous construirons une expression rationnelle $\mathrm{pref}(e)$ telle que $L(\mathrm{pref}(e)) = \mathrm{Pref}(L)$.
3. … par induction structurelle :
4. $\mathrm{pref}(\varnothing) = \varnothing$, $\mathrm{pref}(\varepsilon) = \varepsilon$, $\mathrm{pref}(a) = a + \varepsilon$
5. $\mathrm{pref}(e_1 + e_2) = \mathrm{pref}(e_1) + \mathrm{pref}(e_2)$
   $\mathrm{pref}(e_1 e_2) = \mathrm{pref}(e_1) + e_1\, \mathrm{pref}(e_2)$
   $\mathrm{pref}(e^*) = e^* \mathrm{pref}(e)$ (voir tableau)

Encore une démonstration

Théorème

Si L est un langage rationnel, alors $\mathrm{Pref}(L)$ *l'est aussi.*

Démonstration.

1. Soit e une expression rationnelle telle que $L = L(e)$.
2. Nous construirons une expression rationnelle $\mathrm{pref}(e)$ telle que $L(\mathrm{pref}(e)) = \mathrm{Pref}(L)$.
3. … par induction structurelle :
4. $\mathrm{pref}(\emptyset) = \emptyset$, $\mathrm{pref}(\varepsilon) = \varepsilon$, $\mathrm{pref}(a) = a + \varepsilon$
5. $$\begin{aligned} \mathrm{pref}(e_1 + e_2) &= \mathrm{pref}(e_1) + \mathrm{pref}(e_2) \\ \mathrm{pref}(e_1 e_2) &= \mathrm{pref}(e_1) + e_1\, \mathrm{pref}(e_2) \\ \mathrm{pref}(e^*) &= e^* \mathrm{pref}(e) \qquad \text{(voir tableau)} \end{aligned}$$
6. Maintenant il faut démontrer que, en fait, $L(\mathrm{pref}(e)) = \mathrm{Pref}(L)$.
7. … par induction structurelle, encore (sur tableau).

A vous de jouer

1. Sur alphabet $\Sigma = \{a, b, c\}$, donnez des expressions rationnelles pour les langages suivants :
   1. $\{w \in \Sigma^* \mid w \text{ contient au moins trois } a\}$
   2. $\{w \in \Sigma^* \mid w \text{ contient un nombre pair de } a\}$
   3. $\{w \in \Sigma^* \mid w \text{ contient exactement 2 ou 3 fois le symbole } a\}$
   4. $\{w \in \Sigma^* \mid w \text{ ne contient pas le facteur } cb\}$
2. Sur alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -, .\}$, donnez des expressions rationnelles pour les langages suivants :
   1. les entiers positifs en base 10
   2. les entiers relatifs en base 10
   3. les nombres décimaux positifs en base 10
   4. les nombres décimaux relatifs en base 10
3. Sur alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, -, ., \mathrm{E}\}$, donnez des expressions rationnelles pour les nombres décimaux relatifs en base 10 en notation scientifique.


That's all Folks!