Équivalence des langages rationnels et reconnaissables

Quentin Fortier

January 20, 2024

Équivalence des langages rationnels et reconnaissables

L'objectif de ce cours est de montrer :

Théorème

Soit L un langage. Alors :

$$L \text{ est régulier} \iff L \text{ est reconnaissable}$$

Régulier $\Longrightarrow$ reconnaissable

Preuve de « régulier $\Longrightarrow$ reconnaissable » avec l'**algorithme de Berry-Sethi** :

1. Une expresson régulière e peut être linéarisée (chaque lettre n'est alors utilisée qu'une seule fois).

Régulier $\Longrightarrow$ reconnaissable

Preuve de « régulier $\Longrightarrow$ reconnaissable » avec l'**algorithme de Berry-Sethi** :

1. Une expresson régulière e peut être linéarisée (chaque lettre n'est alors utilisée qu'une seule fois).
2. Un langage linéaire est local.

Régulier $\Longrightarrow$ reconnaissable

Preuve de « régulier $\Longrightarrow$ reconnaissable » avec l'**algorithme de Berry-Sethi** :

1. Une expresson régulière e peut être linéarisée (chaque lettre n'est alors utilisée qu'une seule fois).
2. Un langage linéaire est local.
3. Un langage local est reconnu par un automate local.

Régulier $\Longrightarrow$ reconnaissable

Preuve de « régulier $\Longrightarrow$ reconnaissable » avec l'**algorithme de Berry-Sethi** :

1. Une expresson régulière e peut être linéarisée (chaque lettre n'est alors utilisée qu'une seule fois).
2. Un langage linéaire est local.
3. Un langage local est reconnu par un automate local.
4. Cet automate local peut être « délinéarisé » pour reconnaître L.

L'automate obtenu est appelé **automate de Glushkov**.

Régulier $\Longrightarrow$ reconnaissable : Langage linéaire

Définition

Une expression régulière est **linéaire** si chaque lettre y apparaît au plus une fois.

Régulier $\Longrightarrow$ reconnaissable : Langage linéaire

Définition

Une expression régulière est **linéaire** si chaque lettre y apparaît au plus une fois.

Définition

Soit e une expression régulière sur un alphabet Σ.
Soit k le nombre de lettres (avec multiplicité) apparaissant dans e.
Soit Σ' un alphabet de taille k.
Linéariser e consiste à remplacer chaque occurrence de lettre apparaissant dans e par une lettre différente de Σ'.

Régulier $\Longrightarrow$ reconnaissable : Langage linéaire

Définition

Une expression régulière est **linéaire** si chaque lettre y apparaît au plus une fois.

Définition

Soit e une expression régulière sur un alphabet Σ.
Soit k le nombre de lettres (avec multiplicité) apparaissant dans e.
Soit Σ' un alphabet de taille k.
Linéariser e consiste à remplacer chaque occurrence de lettre apparaissant dans e par une lettre différente de Σ'.

Exemple : soit $e = \varepsilon + b(a + bb)^* b$. En prenant $\Sigma' = \{c_0, c_1, c_2, c_3, c_4\}$, on peut linéariser e en $e' = \varepsilon + c_0(c_1 + c_2 c_3)^* c_4$.

Régulier $\Longrightarrow$ reconnaissable : Langage local

Définition

Soit L un langage. On définit :

- $P(L) = \{a \in \Sigma \mid a\Sigma^* \cap L \neq \emptyset\}$ (premières lettres des mots de L)
- $S(L) = \{a \in \Sigma \mid \Sigma^* a \cap L \neq \emptyset\}$ (dernières lettres des mots de L)
- $F(L) = \{u \in \Sigma^2 \mid \Sigma^* u \Sigma^* \cap L \neq \emptyset\}$ (facteurs de longueur 2 des mots de L)

Régulier $\implies$ reconnaissable : Langage local

Définition

Soit L un langage. On définit :

- $P(L) = \{a \in \Sigma \mid a\Sigma^* \cap L \neq \emptyset\}$ (premières lettres des mots de L)
- $S(L) = \{a \in \Sigma \mid \Sigma^* a \cap L \neq \emptyset\}$ (dernières lettres des mots de L)
- $F(L) = \{u \in \Sigma^2 \mid \Sigma^* u \Sigma^* \cap L \neq \emptyset\}$ (facteurs de longueur 2 des mots de L)

Question

Donner $P(L)$, $S(L)$, $F(L)$ pour $L = a^* b (ab)^* c$.

Régulier $\Longrightarrow$ reconnaissable : Langage local

Définition

Soit L un langage. On définit :

- $P(L) = \{a \in \Sigma \mid a\Sigma^* \cap L \neq \emptyset\}$ (premières lettres des mots de L)
- $S(L) = \{a \in \Sigma \mid \Sigma^* a \cap L \neq \emptyset\}$ (dernières lettres des mots de L)
- $F(L) = \{u \in \Sigma^2 \mid \Sigma^* u \Sigma^* \cap L \neq \emptyset\}$ (facteurs de longueur 2 des mots de L)

Question

Écrire des fonctions `prefixe`, `suffixe`, `facteur` de type `'a regexp -> 'a list` pour déterminer $P(L)$, $S(L)$, $F(L)$.

Régulier $\Longrightarrow$ reconnaissable : Langage local

Définition

Soit L un langage. On définit :

- $P(L) = \{a \in \Sigma \mid a\Sigma^* \cap L \neq \emptyset\}$ (premières lettres des mots de L)
- $S(L) = \{a \in \Sigma \mid \Sigma^* a \cap L \neq \emptyset\}$ (dernières lettres des mots de L)
- $F(L) = \{u \in \Sigma^2 \mid \Sigma^* u \Sigma^* \cap L \neq \emptyset\}$ (facteurs de longueur 2 des mots de L)

Définition

Un langage L est **local** si, pour tout mot $u = u_1 u_2 ... u_n \neq \varepsilon$:

$$u \in L \iff u_1 \in P(L) \wedge u_n \in S(L) \wedge \forall k, u_k u_{k+1} \in F(L)$$

<u>Remarque</u> : $\Longrightarrow$ est évident donc il suffit de prouver $\Longleftarrow$.

Régulier $\Longrightarrow$ reconnaissable : Langage local

Définition

Soit L un langage. On définit :

- $P(L) = \{a \in \Sigma \mid a\Sigma^* \cap L \neq \emptyset\}$ (premières lettres des mots de L)
- $S(L) = \{a \in \Sigma \mid \Sigma^* a \cap L \neq \emptyset\}$ (dernières lettres des mots de L)
- $F(L) = \{u \in \Sigma^2 \mid \Sigma^* u \Sigma^* \cap L \neq \emptyset\}$ (facteurs de longueur 2 des mots de L)

Définition

Un langage L est **local** si, pour tout mot $u = u_1 u_2 ... u_n \neq \varepsilon$:

$$u \in L \iff u_1 \in P(L) \wedge u_n \in S(L) \wedge \forall k, u_k u_{k+1} \in F(L)$$

Remarque : $\Longrightarrow$ est évident donc il suffit de prouver $\Longleftarrow$.

Exercice

Dire si les langages suivants sont locaux : $L_1 = a^*$, $L_2 = (ab)^*$, $L_3 = a^* + (ab)^*$, $L_4 = a^*(ab)^*$.

Régulier $\Longrightarrow$ reconnaissable : Linéaire $\Longrightarrow$ local

Lemme

Soient L_1 et L_2 des langages locaux sur des alphabets disjoints Σ_1 et Σ_2. Alors :

- $L_1 \cup L_2$ est local sur $\Sigma_1 \cup \Sigma_2$
- $L_1 L_2$ est local sur $\Sigma_1 \cup \Sigma_2$
- L_1^* est local sur Σ_1

Théorème

Tout langage linéaire est local.

Preuve : en TD.

Régulier $\implies$ reconnaissable : Automate local

Définition

Un automate déterministe (Σ, Q, q_0, F, E) est **local** si toutes les transitions étiquetées par une même lettre aboutissent au même état :

$$(q_1, a, q_2) \in E \wedge (q_3, a, q_4) \in E \implies q_2 = q_4$$

Régulier $\implies$ reconnaissable : Automate local

Définition

Un automate déterministe (Σ, Q, q_0, F, E) est **local** si toutes les transitions étiquetées par une même lettre aboutissent au même état :

$$(q_1, a, q_2) \in E \wedge (q_3, a, q_4) \in E \implies q_2 = q_4$$

Théorème

Tout langage local L est reconnu par un automate local.

Régulier $\implies$ reconnaissable : Automate local

Définition

Un automate déterministe (Σ, Q, q_0, F, E) est **local** si toutes les transitions étiquetées par une même lettre aboutissent au même état :

$$(q_1, a, q_2) \in E \land (q_3, a, q_4) \in E \implies q_2 = q_4$$

Théorème

Tout langage local L est reconnu par un automate local.

Preuve :

L est reconnu par (Σ, Q, q_0, F, E) où :

- $Q = \Sigma \cup \{q_0\}$: un état correspond à la dernière lettre lue
- $F = S(L)$ si $\varepsilon \notin L$, sinon $F = S(L) \cup \{q_0\}$.
- $E = \{(q_0, a, a) \mid a \in P(L)\} \cup \{(a, b, b) \mid ab \in F(L)\}$

Régulier $\implies$ reconnaissable : Automate local

Définition

Un automate déterministe (Σ, Q, q_0, F, E) est **local** si toutes les transitions étiquetées par une même lettre aboutissent au même état :

$$(q_1, a, q_2) \in E \wedge (q_3, a, q_4) \in E \implies q_2 = q_4$$

Théorème

Tout langage local L est reconnu par un automate local.

<u>Preuve</u> :
L est reconnu par (Σ, Q, q_0, F, E) où :

- $Q = \Sigma \cup \{q_0\}$: un état correspond à la dernière lettre lue
- $F = S(L)$ si $\varepsilon \notin L$, sinon $F = S(L) \cup \{q_0\}$.
- $E = \{(q_0, a, a) \mid a \in P(L)\} \cup \{(a, b, b) \mid ab \in F(L)\}$

<u>Exemple</u> : construire un automate local reconnaissant $e = a(a+b)^*$.

Régulier $\Longrightarrow$ reconnaissable : Algorithme de Berry-Sethi

Soit e une expression régulière.

1. On linéarise e en e', en remplaçant chaque lettre de e par une nouvelle lettre.
2. On construit un automate local A reconnaissant $L(e')$. Pour cela il faut calculer $P(L(e'))$, $S(L(e'))$, $F(L(e'))$.
3. On remplace chaque étiquette a de A en faisant l'opération inverse de 1. On obtient alors un automate (**automate de Glushkov**) reconnaissant $L(e)$.

Régulier $\Longrightarrow$ reconnaissable : Algorithme de Berry-Sethi

Soit e une expression régulière.

1. On linéarise e en e', en remplaçant chaque lettre de e par une nouvelle lettre.
2. On construit un automate local A reconnaissant $L(e')$. Pour cela il faut calculer $P(L(e'))$, $S(L(e'))$, $F(L(e'))$.
3. On remplace chaque étiquette a de A en faisant l'opération inverse de 1. On obtient alors un automate (**automate de Glushkov**) reconnaissant $L(e)$.

On en déduit :

Théorème

L est un langage régulier $\Longrightarrow$ L est reconnaissable.

Régulier $\Longrightarrow$ reconnaissable : Algorithme de Berry-Sethi

Soit e une expression régulière.

1. On linéarise e en e', en remplaçant chaque lettre de e par une nouvelle lettre.
2. On construit un automate local A reconnaissant $L(e')$. Pour cela il faut calculer $P(L(e'))$, $S(L(e'))$, $F(L(e'))$.
3. On remplace chaque étiquette a de A en faisant l'opération inverse de 1. On obtient alors un automate (**automate de Glushkov**) reconnaissant $L(e)$.

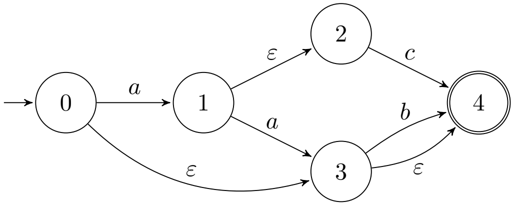
Exercice

Construire l'automate de Glushkov reconnaissant $L(a(a+b)^*)$.

Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

On peut généraliser la notion d'automate en autorisant des ε-transitions (transition étiquetée par ε).

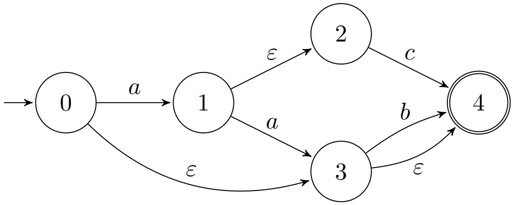
Exemple :



Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

On peut généraliser la notion d'automate en autorisant des ε-transitions (transition étiquetée par ε).

Exemple :



Automate avec ε-transitions reconnaissant
$\{\varepsilon, b, ac, aab\}$

L'**automate de Thompson** utilise des ε-transitions pour obtenir un automate à partir d'une expression régulière.

Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

Théorème

Tout automate avec ε-transitions est équivalent à un automate sans ε-transitions.

Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

Théorème

Tout automate avec ε-transitions est équivalent à un automate sans ε-transitions.

Preuve :

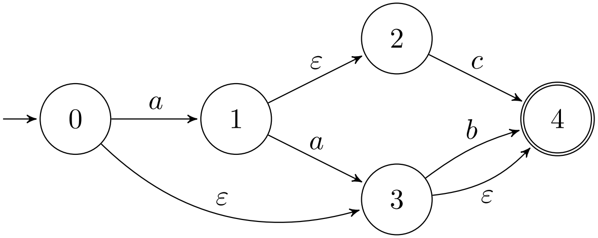
Étant donné un automate avec ε-transitions $A = (\Sigma, Q, I, F, \delta)$, on définit $A' = (\Sigma, Q, I', F, \delta')$ où :

- I' est l'ensemble des états atteignables depuis un état de I en utilisant uniquement des ε-transitions.
- $\delta'(q, a)$ est l'ensemble des états q' tel qu'il existe un chemin de q à q' dans A étiqueté par un a et un nombre quelconque de ε (ce qui peut être trouvé par un parcours de graphe).

Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

Exercice

Donner un automate sans ε-transition équivalent à l'automate suivant :



Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

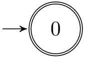
On construit récursivement l'**automate de Thompson** $T(e)$ reconnaissant une expression régulière e.

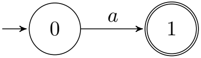
- Cas de base :

Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

On construit récursivement l'**automate de Thompson** $T(e)$ reconnaissant une expression régulière e.

- Cas de base :





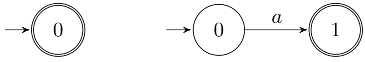
$T(\emptyset)$ $T(\varepsilon)$ $T(a)$

- $T(e_1\, e_2)$:

Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

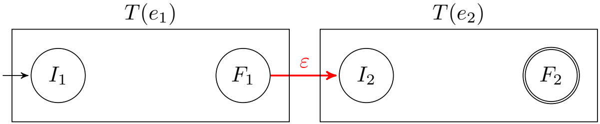
On construit récursivement l'**automate de Thompson** $T(e)$ reconnaissant une expression régulière e.

- Cas de base :



$T(\emptyset)$ $\qquad$ $T(\varepsilon)$ $\qquad$ $T(a)$

- $T(e_1\,e_2)$: on ajoute une ε-transition depuis chaque état final de $T(e_1)$ vers chaque état initial de $T(e_2)$.

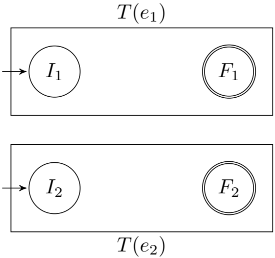


Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

- $T(e_1|e_2)$:

Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

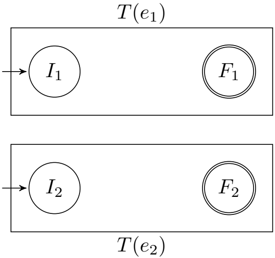
- $T(e_1|e_2)$: on prend l'union des états initiaux et des états finaux.



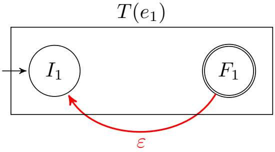
- $T(e_1^*)$:

Régulier $\Longrightarrow$ reconnaissable : Automate de Thompson

- $T(e_1|e_2)$: on prend l'union des états initiaux et des états finaux.



- $T(e_1^*)$: on ajoute une ε-transition depuis chaque état final vers chaque état initial.



Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

Soit L un langage reconnu par un automate A.
La **méthode d'élimination des états** permet de trouver une expression régulière e dont le langage est L.
Pour cela, on considère des automates généralisés où l'étiquette d'une transition peut être une expression régulière, et pas seulement une lettre ou ε.

Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

On commence par se ramener à un automate plus simple :

Lemme

Soit A un automate.
Il existe un automate A' équivalent à A avec :

- Un unique état initial sans transition entrante.
- Un unique état final sans transition sortante.

Preuve :

Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

On commence par se ramener à un automate plus simple :

Lemme

Soit A un automate.
Il existe un automate A' équivalent à A avec :

- Un unique état initial sans transition entrante.
- Un unique état final sans transition sortante.

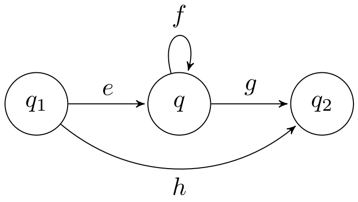
Preuve : On ajoute un état initial q_i et un état final q_f et on ajoute des transitions ε depuis q_i vers les états initiaux de A et depuis les états finaux de A vers q_f.

Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

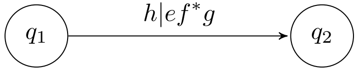
Soit A' un automate comme dans le lemme précédent.
Tant que A' possède au moins 3 états, on choisit un état q différent de q_i et q_f

Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

Soit A' un automate comme dans le lemme précédent.
Tant que A' possède au moins 3 états, on choisit un état q différent de q_i et q_f, on le supprime et on remplace chaque configuration :

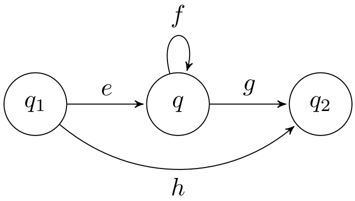


Par :

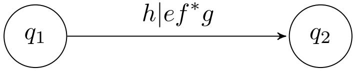


Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

Soit A' un automate comme dans le lemme précédent.
Tant que A' possède au moins 3 états, on choisit un état q différent de q_i et q_f, on le supprime et on remplace chaque configuration :



Par :

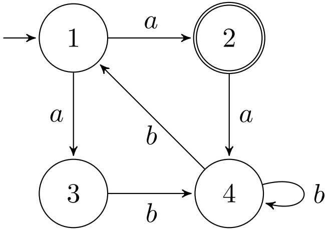


À la fin, l'étiquette de la transition restante donne une expression régulière de même langage que A'.

Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

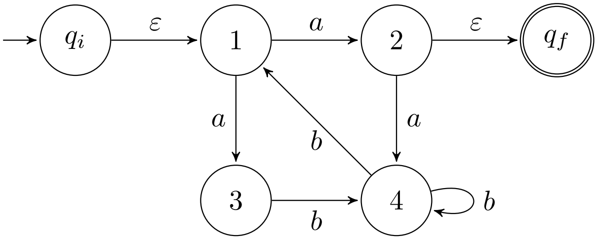
Exercice

Donner une expression régulière de même langage que l'automate suivant, par la méthode d'élimination des états.



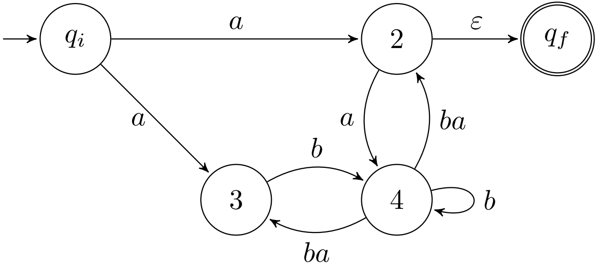
Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

On commence par se ramener à un automate avec un état initial sans transition entrante et un état final sans transition sortante :



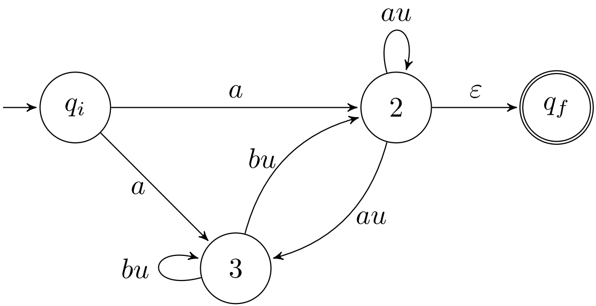
Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

Suppression de l'état 1 :



Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

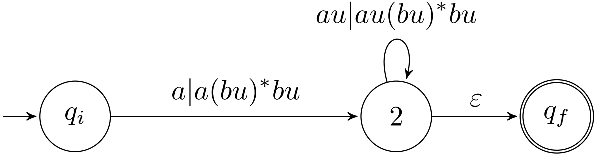
Suppression de l'état 4 :



Avec $u = b^*ba$.

Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

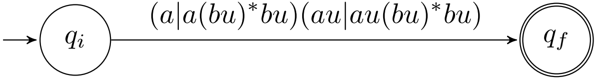
Suppression de l'état 3 :



Avec $u = b^*ba$.

Reconnaissable $\Longrightarrow$ régulier : Élimination d'états

Suppression de l'état 2 :



On obtient l'expression régulière $a|a(bu)^*bu(au|au(bu)^*bu)$ (que l'on peut simplifier), où $u = b^*ba$.

Régulier $\Longleftrightarrow$ reconnaissable

On a donc :

Théorème

Un langage est reconnaissable (par un automate) si et seulement s'il est régulier (décrit par une expression régulière).

Régulier $\Longleftrightarrow$ reconnaissable

On a donc :

Théorème

Un langage est reconnaissable (par un automate) si et seulement s'il est régulier (décrit par une expression régulière).

Tous les théorèmes sur les langages reconnaissables sont donc vraies aussi pour les langages rationnels :

Théorème

Les langages rationnels sont stables par :

- Concaténation, union finie, étoile (par définition).
- Intersection finie, complémentaire, différence (d'après les résultats correspondants sur les automates).

Régulier $\Longleftrightarrow$ reconnaissable

On a donc :

Théorème

Un langage est reconnaissable (par un automate) si et seulement s'il est régulier (décrit par une expression régulière).

Tous les théorèmes sur les langages reconnaissables sont donc vraies aussi pour les langages rationnels :

Théorème

Les langages rationnels sont stables par :

- Concaténation, union finie, étoile (par définition).
- Intersection finie, complémentaire, différence (d'après les résultats correspondants sur les automates).

Exercice

Montrer que l'ensemble des mots sur $\Sigma = \{a, b\}$ ne contenant pas de facteur $ababb$ est régulier.