Dans ce cours, on veut montrer le théorème de Kleene : un langage est régulier si et seulement s'il est reconnaissable.

I Régulier $\implies$ reconnaissable

I.1 Langage linéaire

Définition : Expression régulière linéaire

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

Exemple : $ba|a$ n'est pas linéaire mais a^*b est linéaire.

Définition : Linéarisation

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 $e' = \varepsilon|c_0(c_1|c_2c_3)^*c_4$.

I.2 Langage local

Définition : Linéarisation

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)

Exercice 1.
Donner $P(L)$, $S(L)$, $F(L)$ pour $L = a^*b(ab)^*c$.

Définition : Langage local

Un langage L est local si, pour tout mot $u = u_1u_2...u_n \neq \varepsilon$:

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

Définition équivalente :

Définition : Langage local

Un langage L est local si :

$$L\backslash\{\varepsilon\} = P(L)\Sigma^* \cap \Sigma^*S(L)\backslash\Sigma^*N(L)\Sigma^*$$

où $N(L) = \Sigma^2\backslash F(L)$.

Théorème

Soit L un langage. Si L est local alors L est reconnaissable.

Preuve :

Exercice 2.
Dire si les langages suivants sont locaux :

1. $L_1 = a^*$
2. $L_2 = (ab)^*$
3. $L_3 = a^*|(ab)^*$
4. $L_4 = a^*(ab)^*$

Théorème

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

Preuve :

Attention : Comme vu précédemment, $L_1 = a^*$ et $L_2 = (ab)^*$ sont locaux mais pas $L_3 = L_1 \cup L_2$.

On en déduit :

Théorème

Tout langage linéaire est local.

Preuve :

I.3 Automate local

Définition : Langage local

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.

Preuve :

Exercice 3.
Construire un automate local reconnaissant $e = a(a|b)^*$.

I.4 Algorithme de Berry-Sethi ♡

Algorithme de Berry-Sethi

Entrée : Une expression régulière e.
Sortie : Un automate reconnaissant e, appelé automate de Glushkov.

Linéariser e en e', en remplaçant chaque lettre a de e par une nouvelle lettre $\varphi(a)$.
Calculer $P(L(e'))$, $S(L(e'))$, $F(L(e'))$.
Construire un automate local A reconnaissant $L(e')$.
Renvoyer A dans lequel on a remplacé chaque étiquette a par $\varphi^{-1}(a)$.

Exercice 4.
Appliquer l'algorithme de Berry-Sethi à l'expression régulière $e = a(a|b)^*$.

I.5 ε-transitions

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

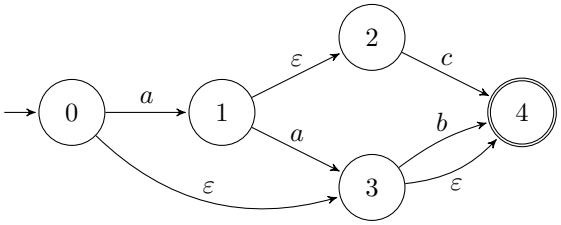
Remarque : un automate avec ε-transitions n'est pas déterministe.

Théorème

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

Preuve :

Exercice 5.
Donner un automate sans ε-transition équivalent à l'automate suivant :



I.6 Autre méthode : automate de Thompson (HP)

La construction de Thompson est une alternative à l'algorithme de Berry-Sethi pour obtenir un automate à partir d'une expression régulière (et donc prouver régulier $\implies$ reconnaissable).

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

- Cas de base :

- $T(e_1e_2)$:

- $T(e_1|e_2)$:

- $T(e^*)$:

Exercice 6.
Construire l'automate de Thompson reconnaissant l'expression régulière $e = a(ab|b)^*$.

II Reconnaissable $\Longrightarrow$ régulier

II.1 Méthode d'élimination des états

Théorème

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 :

Dans l'algorithme suivant, on autorise n'importe quelle expression régulière et on utilise la transformation suivante qui ne change pas le langage reconnu par l'automate :

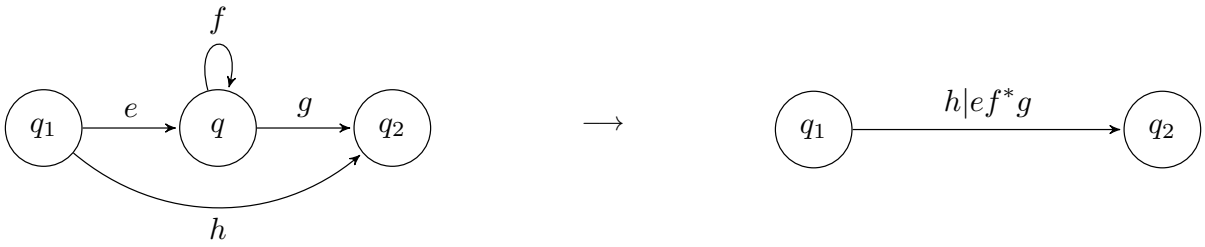


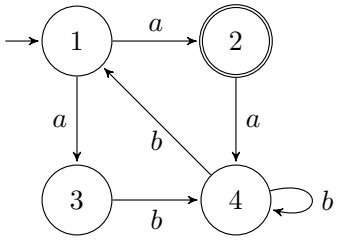
Figure 1: Élimination de l'état q

Algorithme d'élimination des états

Entrée : Un automate A.
Sortie : Une expression régulière e telle que $L(e) = L(A)$.

Construire A' équivalent à A avec un unique état initial q_i et un unique état final q_f.
Tant que A' a au moins 3 états **:**
  Choisir un état q différent de q_i et q_f.
  Éliminer q comme sur la figure 1.
Renvoyer l'étiquette de la transition entre q_i et q_f.

Exercice 7.
Donner une expression régulière de même langage que l'automate suivant, par la méthode d'élimination des états.



III Théorème de Kleene et conséquences

Théorème : Théorème de Kleene

Un langage est régulier si et seulement s'il est reconnaissable.

Conséquences :

- On pourra choisir de raisonner avec des automates ou des expressions régulières, selon ce qui est le plus simple.
- Les langages réguliers sont stables par union, concaténation, étoile, intersection, complémentaire, différence.
- En pratique, il est utile de passer d'une expression régulière (qu'on peut rentrer facilement en ligne de commande) à un automate plus efficace algorithmiquement. La réciproque est moins utile, mais reste un résultat théorique important.