Grammaires et Analyse Syntaxique - Cours 10
Propriétés des langages algébriques

Ralf Treinen

INFORMATIQUE
Sciences
Université Paris Cité

INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

treinen@irif.fr

8 avril 2022

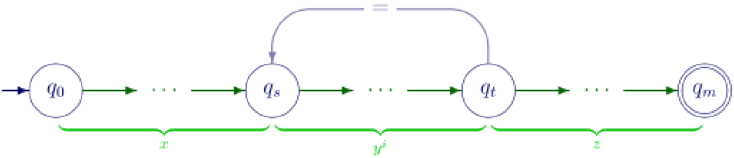
Montrer qu'un langage n'est pas algébrique

- Rappel : un langage L est *algébrique* s'il existe une grammaire algébrique G avec $L = \mathcal{L}(G)$.
- Donc pour montrer qu'un langage est algébrique il suffit de trouver une grammaire.
- Comment faire pour montrer qu'un langage n'est *pas* algébrique ?
- Il nous faut une méthode pour ça quand on veut exploiter les limites des grammaires.

Souvenirs du cours AAL3 : le lemme d'itération

- On a vu en AAL3 une méthode pour montrer qu'un langage n'est pas reconnaissable/régulier : le *lemme d'itération* (ou lemme d'étoile, *pumping lemma*) :
- Pour tout langage régulier L il existe un $N \geq 1$ tel que pour tout $u \in L$ avec $|u| \geq N$: $u = u_1 u_2 u_3$ et
  1. $u_2 \neq \epsilon$
  2. $|u_1 u_2| \leq N$
  3. $u_1 u_2^i u_3 \in L$ pour tout $i \geq 0$
- Intuitivement : la capacité de mémorisation d'un automate est limitée par son nombre d'états.

Illustration du lemme d'itération des langages réguliers

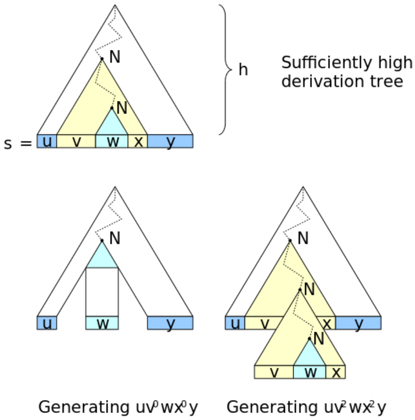


Source : Jochen Burghardt/wikipedia

Des langages réguliers aux langages algébriques

- Le lemme d'itération des langages réguliers vient des automates : Si un mot d'entrée est suffisamment long alors il y un état de l'automate qui se répète. On peut alors itérer le mot lu entre ces deux occurrences du même état.
- Pour les langages algébriques on peut obtenir un lemme similaire, basé sur les arbres de dérivation : si un arbre de dérivation est suffisamment profond alors il y a non-terminal de la grammaire qui se répète sur une branche de cet arbre.
- On peut alors itérer le mot produit "entre ces deux occurrences du même non-terminal" (voir le dessin).

Idée de la preuve du lemme d'itération



Source : Jochen Burghardt/wikipedia

Un lemme d'itération pour les langages algébriques

Lemme d'itération pour les langages algébriques

Soit L un langage algébrique. Alors il existe un $p \geq 1$ tel que tout mot $s \in L$ avec $|s| \geq p$ peut être décomposé en $s = uvwxy$ avec :

- $|vwx| \leq p$
- $|vx| \geq 1$
- $uv^iwx^iy \in L$ pour tout $i \geq 0$

Idée de la preuve du lemme d'itération

- Donnée une grammaire $G = (\Sigma, N, S, P)$.
- Soit $h = card(N) + 1$.
- Soit $k = max\{|\alpha| \mid (N \to \alpha) \in P\}$, c-a-d la longueur maximale des côtés droits des productions.
- Un arbre de dérivation d'hauteur h peut seulement produire des mots de longueur $\leq k^{h-1}$.
- Si $w \in \mathcal{L}(G)$ avec $|w| > k^{h-1}$ alors l'arbre de dérivation de w a une branche sur laquelle un non-terminal parait deux fois.
- Certains détails qui sont omis ici.

Négation de la propriété d'itération

- Donc on peut montrer qu'un langage L n'est *pas* algébrique si on montre que L satisfait la *négation* de la propriété d'itération.
- C'est-à-dire il faut montrer :
  - $\forall p$ avec $p \geq 1$:
  - $\exists s \in L$ avec $|s| \geq p$
  - $\forall u, v, w, x, y$ avec $s = uvwxy$, $|vwx| \leq p$, $|vx| \geq 1$
  - $\exists i$ tel que $uv^{i}wx^{i}y \notin L$
- Comment faire une telle preuve ?

Utiliser le lemme d'itération

- On peut voir une telle formule logique avec une alternance de quantificateurs $\forall$ et $\exists$ comme un jeu entre deux joueurs : Existentiel et Universel.
- Existentiel cherche à montrer que la formule est vraie, Universel cherche à l'empêcher.
- Les deux choisissent des valeurs de variables : Existentiel pour les variables existentielles, Universel pour les variables universelles.
- Les joueurs peuvent prendre en considération les choix précédents de leur adversaire.
- La formule est vrai si Existentiel a une *stratégie gagnante*.

Application : un langage qui n'est pas algébrique

- Nous allons montrer : $L = \{a^n b^n c^n \mid n \geq 0\}$ n'est pas algébrique.
- L'opposant choisi une valeur $p \geq 1$.
- Nous choisissons $s = a^p b^p c^p$.
- L'opposant choisi une décomposition $uvwxy = a^p b^p c^p$ avec $|vwx| \leq p$ et $|vx| \geq 1$.
- Il est maintenant à nous de choisir i, et de démontrer que $uv^i wx^i y \notin L$. On considère les cas différents ou dans s se trouve la partie vwx :

Application : un langage qui n'est pas algébrique

- Si $vwx \in a^+$: il existe k, l avec $v = a^k$ et $x = a^l$ et $k + l > 0$. Nous choisissons $i = 0$, on a alors $uwy = a^{p-k-l} b^p c^p \notin L$ car $p - k - l \neq p$
- Si $vwx \in b^+$: pareil
- Si $vwx \in c^+$: pareil
- Si $vwx \in a^+ b^+$: on a que v ou x contiennent au moins un a ou un b, mais pas de c. Nous choisissons $i = 0$, on a alors $|uwy|_a < |uvwxy|_a$ ou $|uwy|_b < |uvwxy|_b$, mais $|uwy|_c = |uvwxy|_c$. Donc $uwy \notin L$.
- Si $vwx \in b^+ c^+$: pareil
- Si $vwx \in a^+ b^+ c^+$: pas possible car $|vwx| \leq p$.

Clôture sous intersection avec des langages régulier

- Nous avons vu la semaine dernière : la classe des langages algébriques est close sous intersection avec des langages réguliers, c'est-à-dire si L_1 algébrique et L_2 régulier, alors $L_1 \cap L_2$ algébrique.
- Preuve par construction d'un automate produit entre un automate à pile pour L_1 et un automate fini pour L_2.
- On peut s'en servir pour montrer qu'un langage n'est pas algébrique, similaire à la façon comment nous avons exploité les propriétés des langages réguliers en L2.

Exemple

- Soit
$$L = \{a^j b^k c^l d^m \mid j = 0 \text{ ou } k = l = m\}$$
- Supposons que L soit algébrique. Alors, $L \cap \mathcal{L}(ab^*c^*d^*)$ est également algébrique (par la clôturé des algébriques avec des langages réguliers, voir le cours de la semaine dernière).
- Or,
$$\begin{aligned} & \{a^j b^k c^l d^m \mid j = 0 \text{ ou } k = l = m \geq 0\} \cap \mathcal{L}(ab^*c^*d^*) \\ = \; & \{ab^k c^k d^k \mid k \geq 0\} \end{aligned}$$

  et on peut montrer, comme sur l'exemple précédent, que ce langage n'est pas algébrique.
- Absurde, donc L n'est pas algébrique.

Exemple (suite)

- Le langage L satisfait la propriété du lemme d'itération !
- Nous choisissons $p = 1$
- Cas 1 : l'adversaire choisi un mot $s = b^k c^l d^m$ de longueur ≥ 1.
  - On a donc $k + l + m \geq 1$, par exemple $l \geq 1$ (les autres cas sont pareil).
  - Nous croissions une décomposition $s = uvwxy$:

$$u = b^k \quad v = c \quad w = \epsilon \quad x = \epsilon \quad y = c^{l-1} d^m$$

  - Pour n'importe quel i :

$$uv^i wx^i y = b^k c^{l-1+i} d^m \in L$$

Exemple (suite)

- Cas 2 : l'adversaire choisi un mot $s = a^j b^k c^k d^k$ de longueur ≥ 1 et $j \geq 1$:
  - Nous choisissons une décomposition $s = uvwxy$:

$$u = \epsilon \quad v = a \quad w = \epsilon \quad x = \epsilon \quad y = a^{j-1} b^k c^k d^k$$

  - Pour n'importe quel i :

$$uv^i wx^i y = a^{j-1+i} b^k c^k d^k \in L$$

- On voit aussi que le lemme d'itération énonce seulement une propriété *nécessaire* des langages algébriques.
- Donc on peut utiliser le lemme d'itération seulement pour montrer qu'un langage n'est *pas* algébrique.

Propriétés de clôture : union

- Les langages algébriques sont clos sous union : Si L_1 et L_2 sont algébriques, alors $L_1 \cup L_2$ est aussi algébrique.
- Preuve : Soient les grammaires

$$\begin{array}{ll} G_1 = (\Sigma_1, N_1, S_1, P_1) & \text{avec } L_1 = \mathcal{L}(G_1) \\ G_2 = (\Sigma_2, N_2, S_2, P_2) & \text{avec } L_2 = \mathcal{L}(G_1) \end{array}$$

- On suppose que N_1 et N_2 sont disjoints.
- Grammaire pour $L_1 \cup L_2$:

$$(\Sigma_1 \cup \Sigma_2, N_1 \cup N_2 \cup \{S_{new}\}, S_{new}, P_1 \cup P_2 \cup S_{new} \to S_1 \mid S_2)$$

Propriétés de clôture : concaténation

- Les langages algébriques sont clos sous concaténation : Si L_1 et L_2 sont algébriques, alors $L_1 L_2$ est aussi algébrique.
- Preuve : Soient les grammaires

$$\begin{array}{ll} G_1 = (\Sigma_1, N_1, S_1, P_1) & \text{avec } L_1 = \mathcal{L}(G_1) \\ G_2 = (\Sigma_2, N_2, S_2, P_2) & \text{avec } L_2 = \mathcal{L}(G_1) \end{array}$$

- On suppose que N_1 et N_2 sont disjoints.
- Grammaire pour $L_1 L_2$:

$$(\Sigma_1 \cup \Sigma_2, N_1 \cup N_2 \cup \{S_{new}\}, S_{new}, P_1 \cup P_2 \cup S_{new} \rightarrow S_1\, S_2)$$

Propriétés de clôture : intersection

- Les langages algébriques ne sont *pas* clos sous intersection : il existe deux langages algébriques L_1 et L_2 tel que $L_1 \cap L_2$ n'est pas algébrique.
- $L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$
- L_1 est algébrique car concaténation de deux langages algébriques :
  - $\{a^n b^b \mid n \geq 0\}$, grammaire $S \rightarrow aSb \mid \epsilon$;
  - c^*, c'est même un langage rationnel.
- $L_2 = \{a^m b^n c^n \mid n, m \geq 0\}$, algébrique pour la même raison.
- Or, $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ qui n'est pas algébrique.

Propriétés de clôture : complément

- Les langages algébriques ne sont *pas* clos sous complément.
- Preuve :
  - Supposons pour l'absurde que le complément $\overline{L}$ de tout langage algébrique L est aussi algébrique.
  - Il en suit que les langages algébriques sont aussi clos sous intersection, car
    $$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$
  - Contradiction avec le transparent précédent.

Un contre-exemple à la clôture par complément ?

- On aimerait bien voir l'exemple concret d'un langage L qui est algébrique, mais son complément n'est pas algébrique, comme nous avons vu l'exemple de deux langages algébriques dont l'intersection ne l'est pas.
- Est-ce que notre preuve nous permet à extraire cet exemple ?
- Non, malheureusement !

Un contre-exemple à la clôture par complément ?

- On a vu que L_1 et L_2 sont algébriques, mais $\overline{\overline{L_1} \cup \overline{L_2}}$ ne l'est pas.
- Soit $\overline{L_1}$ n'est pas algébrique : L_1 est l'exemple cherché.
- Soit $\overline{L_2}$ n'est pas algébrique : L_2 est l'exemple cherché.
- Soit $\overline{L_1}$ et $\overline{L_2}$ sont algébriques tous les deux, alors leur union l'est aussi, et $\overline{L_1} \cup \overline{L_2}$ est l'exemple cherché.
- Mais on ne sait pas quel cas s'applique !
- Il s'agit d'une preuve *non constructive*.

Un contre-exemple concret à la clôture par complément

- Notre exemple est le langage des carrés :
  $L = \{ww \mid w \in \{a, b\}^*\}$.
- Nous avons vu en L2 que ce langage n'est pas régulier.
- Maintenant nous allons montrer :
  - L n'est pas algébrique
  - $\overline{L}$ est algébrique (ce qui peut surprendre).
- Donc, le contre exemple cherché est le complément de L.

Application du lemme d'itération au langage des carrés

- L'opposant choisi une valeur $p \geq 1$
- Nous choisissons le mot suivant qui est en L :

$$s = \underbrace{a \cdots a}_{p}\underbrace{b \cdots b}_{p}\underbrace{a \cdots a}_{p}\underbrace{b \cdots b}_{p}$$

- Dans le mot s il y a 4 blocs alternants de a et de b, chacun de longueur p.
- L'opposant choisi un découpage $s = uvwxy$, avec $|vwx| \leq p$ et $|vx| \geq 1$.
- Nous choissons $i = 0$, il nous reste à argumenter que $uwy \notin L$.

Application du lemme d'itération au langage des carrés (2)

$$s = \underbrace{a \cdots a}_{p}\underbrace{b \cdots b}_{p}\underbrace{a \cdots a}_{p}\underbrace{b \cdots b}_{p}$$

- Cas 1 : *vwx* est entièrement dans la partie verte ou la partie rouge : contradiction car, après passage à *uwy*, les parties verte et rouge sont de longueur différente !
- Cas 2 : *vwx* est à cheval entre vert et rouge. Dans ce cas *v* contient des *b* verts, ou *x* contient des *b* rouges. (attention un des deux peut être ϵ.)
  Donc, on passant à *uwy*, on efface des b dans le vert, ou des a dans le rouge, ou les deux : contradiction !

Une grammaire pour le complément de L

- Le complément de L consiste en tous les mots qui ne sont *pas* de la forme ww. Pour cela il y a deux possibilités :
  1. il est de longueur impaire ;
  2. il est de longueur paire, et quand on le coupe en deux moitiés $w_1 w_2$, avec $|w_1| = |w_2|$, il y a une position i telle que $w_1[i] \neq w_2[i]$.
- Il suffit donc de montrer que chacun des deux cas donne lieu à un langage algébrique (car les algébriques sont clos sous union).
- Le langage du premier cas est même régulier :

$$((a \mid b)(a \mid b))^*(a \mid b)$$

Une grammaire pour le complément de L

- Pour le langage $L_1 = \{w_1 w_2 \mid |w_1| = |w_2|, w_1 \neq w_2\}$:

$$\begin{aligned} x \in L_1 \quad & \Leftrightarrow \quad \exists y_1, y_2, z_1, z_2 . x = y_1 a z_1 y_2 b z_2, |y_1| = |y_2|, |z_1| = |z_2| \\ & \qquad \text{(ou la même chose avec } a \text{ et } b \text{ inversés)} \\ & \Leftrightarrow \quad \exists y_1, v, z_2 . x = y_1 a v b z_2, |y_1| + |z_2| = |v| \\ & \Leftrightarrow \quad \exists y_1, y_2, z_1, z_2 . x = y_1 a y_2 z_1 b z_2, |y_1| = |y_2|, |z_1| = |z_2| \end{aligned}$$

- Donc on a réussi à inverser y_2 et z_1. Ca change tout car maintenant le problème ressemble à la génération des palindromes !

Une grammaire pour le complément de L

- On peut donc écrire la grammaire pour L_1 :
-

$$
\begin{array}{lcl}
S & \to & \mathsf{A}\ \mathsf{B} \mid \mathsf{B}\ \mathsf{A} \\
\mathsf{A} & \to & \mathsf{L}\ \mathsf{A}\ \mathsf{L} \mid \mathtt{a}
\end{array}
\qquad
\begin{array}{lcl}
\mathsf{L} & \to & \mathtt{a} \mid \mathtt{b} \\
\mathsf{B} & \to & \mathsf{L}\ \mathsf{B}\ \mathsf{L} \mid \mathtt{b}
\end{array}
$$

avec axiome S.

La morale de cet exemple

- Cet exemple montre une limitation intéressante des grammaires : elles ne peuvent pas assurer la propriété qu'on a deux occurrences du même facteur dans un mot, au moins pas quand ce facteur est un mot de longueur non bornée.
- Ca reste vrai quand les deux occurrences sont séparées par des autres symboles.
- C'est très pertinent pour l'analyse des langages informatiques :
  - Dans XML on a des balises ouvrantes `<w>` et fermantes `</w>`, où w est un mot de longueur non bornée.
  - Dans des langages de programmation on exige souvent qu'une variable doit être déclarée avant son utilisation.
- Comment réaliser des telles restrictions dans l'analyse ?

Au delà de l'analyse syntaxique

- La réponse est qu'on ne le fait pas dans l'analyse *syntaxique*, mais dans une phase successive appelée l'analyse *sémantique*.
- Cette analyse sémantique est faite sur l'arbre de syntaxe abstraite produit par l'analyse syntaxique.
- On parle aussi de sémantique *statique* car on peut l'analyser sans d'exécuter le programme.
- Exemple : vérification des types des variables : voir le code exemple.

Exemple

- Vérification de typage d'un langage impératif avec des types int et bool :
  - Absence de double déclaration de la même variable
  - Toute variable utilisée déclarée
  - Expressions correctement typées
  - Instructions correctement typée