Langages formels (11X003) - Automne 2026

9. Lemme de l'étoile hors-contexte (et au-delà)

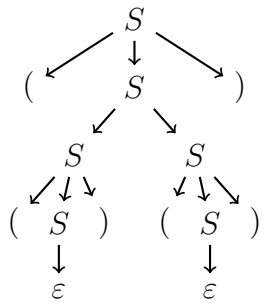
Enseignant: Arnaud Casteigts

Assistants: E. Arm & M. Marseloo
Moniteurs: A. Hugener & O. Oternaud

Nous avons déjà vu comment montrer qu'un langage n'est pas régulier en utilisant le lemme de l'étoile (*c.f.* Cours 5). Dans ce cours, nous allons utiliser une démarche analogue pour les langages hors-contextes, en utilisant une autre version du lemme de l'étoile. Puis nous discuterons de la façon dont les langages hors-contextes se comportent sous les opérations d'union, de concaténation, d'intersection et de complément. Nous finirons par un rapide survol de l'étage suivant, à savoir les langages contextuels (et grammaires contextuelles).

9.1 Lemme de l'étoile hors-contexte (intuition)

Par définition, si un langage L est hors-contexte, alors il peut être engendré par une grammaire hors-contexte G. Chaque mot de L correspond alors à un *arbre de dérivation* de G, comme dans l'exemple ci-contre pour le mot "(()())" et la grammaire $S \to (S) \mid SS \mid \epsilon$, vue dans un cours précédent.



De manière plus générale, étant donné une grammaire G et un mot z de $L(G)$. Si la longueur de z est suffisamment grande, alors pour n'importe quel arbre de dérivation qui lui correspond, au moins une variable doit être *répétée* dans un chemin de la racine vers les feuilles. Cela est illustré ci-dessous pour une grammaire quelconque dans laquelle une variable N serait répétée :

Un tel mot z peut alors être décomposé en facteurs $uvwxy$ (voir le dessin), tels que u (resp. y) correspond à la partie du mot produite à gauche (resp. à droite) de la première occurrence de la variable N, et vwx est la partie dérivée de la première variable N, où v,w,x sont les parties dérivées à gauche, en dessous, et à droite (respectivement) de la deuxième occurrence de N. Puisque la variable N est capable de se reproduire elle-même, elle pourrait également le faire un nombre arbitraire de fois avant de produire w (dessin de droite), ce qui implique que le mot uv^iwx^iz doit aussi faire partie du langage pour tout $i \geq 0$.

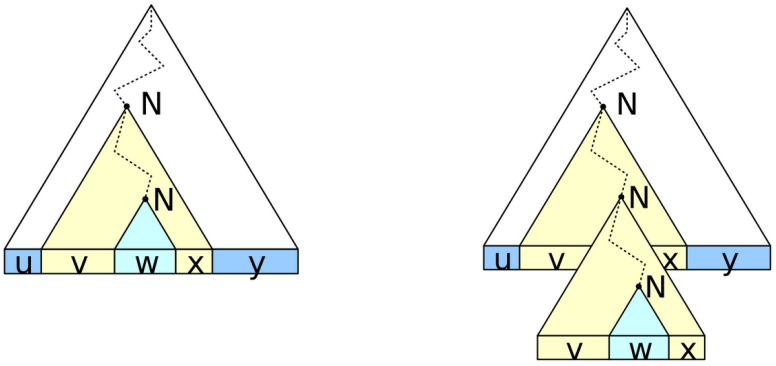


FIGURE 1 – À gauche : arbre de dérivation d'un mot z suffisamment long. À droite : arbre de dérivation d'un mot z' obtenu en répétant le comportement d'une partie de la dérivation.

9.2 Le lemme lui-même

Le lemme de l'étoile pour les langages hors-contexte est similaire à celui des langages réguliers. Il s'agit d'une propriété que tout langage hors-contexte *doit* satisfaire. Le voici :

Lemme 9.1 (Lemme de l'étoile pour les LHC)**.** *Si L est un langage hors-contexte, alors il existe une longueur k au delà de laquelle tout mot $z \in L$ peut s'écrire $z = uvwxy$ avec :*

1. $|v| + |x| > 0$
2. $|vwx| \leq k$
3. $uv^iwx^iy \in L$ *pour tout* $i \in \mathbb{N}$.

Prenons l'exemple du langage $L = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$. Soit k la longueur critique associée à ce langage. Prenons un mot $z \in L$ tel que $|z| > k$, par exemple $z = \mathtt{a}^k\mathtt{b}^k$. On peut décomposer z sous la forme $u \cdot v \cdot w \cdot x \cdot y$ avec $u = \mathtt{a}^{k-1}, v = \mathtt{a}, w = \varepsilon, x = \mathtt{b}$ et $y = \mathtt{b}^{k-1}$. On a bien $|vwx| \leq k$ et $|vx| > 0$. Enfin, il est facile de voir que $uv^iwx^iy \in L$ pour tout $i \in \mathbb{N}$.

L'usage de ce lemme est similaire à celui pour les langages réguliers : si l'on veut montrer qu'un langage L n'est *pas* hors-contexte, il suffit de montrer que pour n'importe quel k, il existe un mot plus long que k qui *ne peut pas être décomposé* sous la forme indiquée. Comme pour l'autre lemme de l'étoile, cela se fait généralement par l'absurde, en supposant que le lemme est satisfait et en obtenant une contradiction.

9.2.1 Exemple

Montrons que le langage $L = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \in \mathbb{N}\}$ sur l'alphabet $\Sigma = \{a, b, c\}$ n'est pas hors-contexte. Par l'absurde, supposons que L est hors-contexte, le lemme nous dit qu'il existe un k tel que tout mot z de longueur $\geq k$ est décomposable en facteurs $uvwxy$ avec les trois propriétés vraies. Nous allons montrer que cela est faux. Quel que soit k, on peut prendre le mot $z = \mathtt{a}^k\mathtt{b}^k\mathtt{c}^k \in L$. Ce mot est censé être décomposable sous la forme $uvwxy$ de sorte que $|v| + |x| \geq 0$ et $|vwx| \leq k$. Il y a cinq cas possibles pour la partie vwx :

- vwx ne contient que des `a`
- vwx ne contient que des `b`
- vwx ne contient que des `c`
- vwx ne contient que des `a` et des `b`
- vwx ne contient que des `b` et des `c`

Observez que vwx ne peut pas contenir les trois types de symboles, car $|vwx| \leq k$, ce qui est trop petit pour cela. Examinons maintenant les conséquences de la répétition des facteurs v et x (propriété 3 du lemme). Si vwx ne contient que des `a`, on obtiendra un mot qui a trop de `a` (donc pas dans L, contradiction). Idem pour `b` et pour `c`. Dans le quatrième cas, le symbole `c` se retrouvera en trop petite quantité par rapport aux autres, et dans le cinquième cas, c'est le symbole `a` qui sera en trop petite quantité. Dans tous les cas, on arrive donc à une contradiction, ce qui implique que z n'est pas décomposable sous la forme spécifiée et donc L n'est pas hors-contexte.

9.3 Remarques générales sur les langages hors-contextes

Si L est un langage hors-contexte (h.c.), alors :

- L^* est h.c.
- $\overline{L}$ n'est pas forcément h.c.

Si L_1 et L_2 sont des langages hors-contextes, alors :

- $L_1 \cup L_2$ est h.c.
- $L_1 \circ L_2$ est h.c.
- $L_1 \cap L_2$ n'est pas forcément h.c.
  Par contre, si l'un des deux est *régulier*, on a la garantie que $L_1 \cap L2$ est h.c.

Les opérations sur les langages hors-contextes sont donc à manipuler avec précaution. Notamment, ils ne sont pas clos par intersection et complémentation.

9.4 Langages contextuels (tour d'horizon rapide)

Nous avons vu que le langage $\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n$ n'est pas hors-contexte (via le lemme de l'étoile pour les LHC). Ce langage appartient à la famille suivante, qui est celle des *langages contextuels* dont les langages hors contextes sont des cas particuliers (un air de déjà vu...).

9.4.1 Grammaire contextuelle

Une grammaire $G = (V, \Sigma, \mathcal{P}, S)$ est **contextuelle** si toutes les règles de production de $\mathcal{P}$ sont de la forme $uXv \to u\,\gamma\,v$, où u et v sont des mots quelconques de $(V \cup \Sigma)^*$ appelés

le *contexte* de cette règle, X est une variable de V et γ est un mot quelconque de $(V \cup \Sigma)^+$. Notez le $^+$ et non le * pour γ, car on interdit γ d'être vide pour interdire que la longueur d'un mot intermédiaire puisse diminuer en cours de dérivation (ce qui rendrait la grammaire trop puissante). Du coup, si l'on veut que ε fasse partie du langage, on ajoute explicitement la règle $S \to \varepsilon$ en plus des autres règles (en interdisant que S soit aussi utilisée dans la partie droite d'une autre règle, sinon on retombe sur le cas où la taille peut diminuer).

Conceptuellement, la principale nouveauté de ces grammaires est que l'*activabilité* d'une règle pour remplacer une variable X peut dépendre de ce qui se trouve autour : son *contexte*. Le contexte lui-même n'a pas le droit de changer, mais cela donne déjà plus de puissance.

Exemple. Le langage $L = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 1\}$ peut être engendré par la grammaire suivante :

1. $S \to \mathtt{a}BC$
2. $S \to \mathtt{a}SBC$
3. $CB \to CZ$
4. $CZ \to WZ$
5. $WZ \to WC$
6. $WC \to BC$
7. $\mathtt{a}B \to \mathtt{ab}$
8. $\mathtt{b}B \to \mathtt{bb}$
9. $\mathtt{b}C \to \mathtt{bc}$
10. $\mathtt{c}C \to \mathtt{cc}$

Les deux premières règles permettent de générer les mots intermédiaires de la forme $\mathtt{a}^n(BC)^n$, par exemple $\mathtt{aaa}BCBCBC$. Notez qu'à ce stade, les variables B et C se retrouvent entrelaçées. Les règles 3 à 6 permettent d'échanger successivement chaque motif CB en BC (ce qui ne peut pas être fait en une seule règle dans le format imposé) ; Enfin, les règles 7 à 10 permettent de remplacer les variables B et C par les symboles terminaux correspondant (`b` et `c`), à condition qu'ils soient à la bonne place (grâce au contexte !).

Par exemple, on peut produire le mot `aabbcc` comme suit :

$S \overset{r_2}{\Longrightarrow} \mathtt{a}SBC \overset{r_1}{\Longrightarrow} \mathtt{aa}BCBC \overset{r_3}{\Longrightarrow} \mathtt{aa}BCZC \overset{r_4}{\Longrightarrow} \mathtt{aa}BWZC \overset{r_5}{\Longrightarrow} \mathtt{aa}BWCC \overset{r_6}{\Longrightarrow} \mathtt{aa}BBCC \overset{r_7}{\Longrightarrow} \mathtt{aab}BCC \overset{r_8}{\Longrightarrow} \mathtt{aabb}CC \overset{r_8}{\Longrightarrow} \mathtt{aabbc}C \overset{r_9}{\Longrightarrow} \mathtt{aabbc}C \overset{r_{10}}{\Longrightarrow} \mathtt{aabbcc}$.

En terme de machine les reconnaissant, les langages contextuels correspondent exactement aux machines de Turing *linéairement bornés*. Ces machines sont un cas particulier de machines de Turing, dont la mémoire est proportionnelle à la longueur du mot d'entrée (le facteur de proportionalité dépend du langage). Nous ne prendrons pas trop de temps à parler de ce modèle intermédiaire et passerons directement aux machines de Turing générales.