Langages formels (11X003) - Automne 2024

9. Lemme de l'étoile (hors-contexte)

Enseignant: Arnaud Casteigts

Assistants: A.-Q. Berger & M. De Francesco
Monitrices: L. Heiniger & A. Tekkoyun

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.

9.1 Rappel du lemme de l'étoile pour les langages réguliers

Le lemme de l'étoile pour les langages *réguliers* établit une propriété que tout langage régulier doit satisfaire. Intuitivement, il établit que tout mot dépassant une certaine longueur doit contenir une partie (un *facteur*) qui peut être répété un nombre arbitraire de fois, tout en restant dans le langage. Nous le rappelons ici, en changeant le nom des variables pour mieux coller à la version d'aujourd'hui.

Lemme 9.1 (Lemme de l'étoile pour les langages réguliers)**.** *Si un langage L est régulier, alors il existe une longueur k au delà de laquelle tout mot $z \in L$ peut s'écrire sous la forme uvw avec :*

1. $|v| > 0$,
2. $|uv| \leq k$,
3. $uv^i w \in L$ *pour tout* $i \in \mathbb{N}$.

Pour montrer qu'un langage L n'est pas régulier, il suffit donc de trouver un mot $z \in L$ tel que pour toute décomposition $z = uvw$, la répétition du facteur v peut nous faire sortir de L. Nous avions donné l'exemple du langage $L = \{a^n b^n \mid n \in \mathbb{N}\}$.

9.2 Lemme de l'étoile pour les langages hors-contextes

Un lemme similaire existe pour les langages hors-contextes, le voici :

Lemme 9.2 (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* $i \in \mathbb{N}$.

En effet, si L est hors-contexte, on peut toujours faire cela. Prenons l'exemple du langage $L = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$ (qui est hors-contexte) et un mot quelconque de L, par exemple $z = \mathtt{aaabbb}$ (en l'occurrence, $k = 2$ pour ce langage). On peut décomposer $z = \mathtt{aa} \cdot \mathtt{a} \cdot \varepsilon \cdot \mathtt{b} \cdot \mathtt{bb}$, autrement dit $u = \mathtt{aa}, v = \mathtt{a}, w = \varepsilon, x = \mathtt{b}$ et $y = \mathtt{bb}$. On a bien $|vwx| \leq k$ et $|vx| > 0$. Enfin, il est facile de voir que $uv^iwx^iy \in L$ pour tout $i \in \mathbb{N}$.

Si L est un langage hors-contexte, alors on doit pouvoir faire cela pour *tous les mots* de L de longueur $\geq k$ (c'est le cas ici). Le lemme peut donc être utilisé dans l'autre sens : si l'on veut montrer qu'un langage L' n'est pas hors-contexte, il suffit de montrer qu'il existe un mot de L' pour lequel cela ne marche pas.

9.2.1 Exemple d'utilisation

Utilisons le lemme de l'étoile pour montrer 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 d'abord 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 ne marche pas. Prenons par exemple le mot $z = \mathtt{a}^k\mathtt{b}^k\mathtt{c}^k \in L$ et décomposons-le de sorte que v et x ne sont pas vides tous les deux (propriété 1 du lemme), il y a quatre possibilités pour la partie vwx :

- vwx ne contient que des `a`
- vwx ne contient que des `b`
- vwx ne contient que des `c`
- vwx contient deux types de symboles : `a` et `b`, ou `b` et `c` (il ne peut pas en contenir trois, car la décomposition satisfait $|vwx| \leq k$ (propriété 2), ce qui est trop petit pour couvrir trois symboles différents dans le mot choisi).

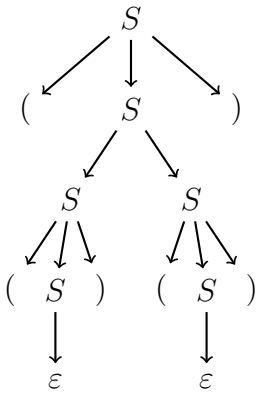
Examinons maintenant les conséquences de la répétition des facteurs v et x (propriété 3 du lemme). Si vwx ne contient que des `a`, on obtiendra un mot qui a trop de `a` (donc pas dans L, contradiction). Idem pour `b` et idem pour `c`. Reste le dernier cas, mais ici le symbole qui n'apparaît pas dans vwx se retrouvera en trop petite quantité. Dans tous les cas, on arrive donc à une contradiction, ce qui implique que L n'est pas hors-contexte.

9.2.2 Démonstration du lemme lui-même

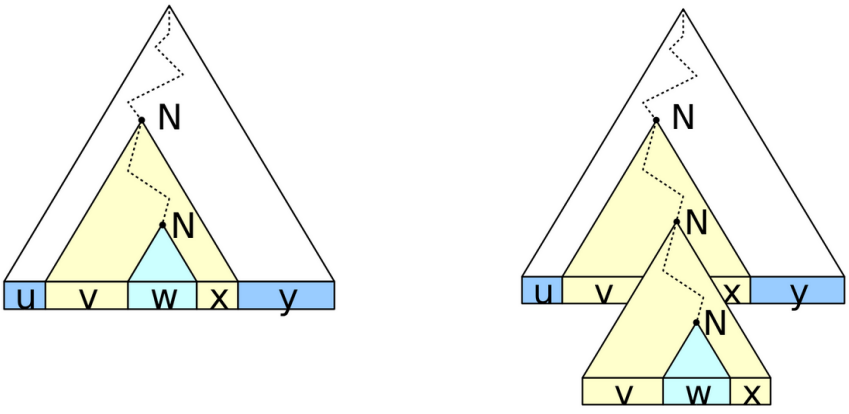
Nous allons maintenant donner les idées principales du lemme lui-même. Il n'est pas nécessaire de connaître la preuve en détail, mais une bonne compréhension de ces idées vous aidera certainement à bien l'utiliser.

Par définition, si un langage L est hors-contexte, alors il peut être engendré par une grammaire hors-contexte G. Si un mot w appartient à L, on peut donc l'obtenir par une dérivation de G et on peut également l'associer à un arbre de dérivation.

Par exemple, pour la grammaire $S \to (S) \mid SS \mid \varepsilon$, qui produit les mots bien parenthésés (*c.f.* Cours 6), le mot "(()())" correspond à l'arbre suivant :



Dans cet exemple, il n'y a qu'une variable dans la grammaire (en l'occurence, S), mais en général, une grammaire hors-contexte peut en avoir plusieurs. L'idée du lemme de l'étoile est la suivante : si un mot z est suffisamment long, 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 (et produire des symboles terminaux entre-temps). C'est clairement le cas dans l'exemple précédent, on peut aussi imaginer un cas plus général, par exemple l'arbre de dérivation ci-dessous (à gauche) pour une grammaire qui aurait plusieurs variables, dont la variable N :



Un mot z du langage peut alors être décomposé en facteurs $uvwxy$ tels que u correspond à la partie du mot produite à gauche de la première variable N (idem pour y à droite), et vwx est la partie qui dérive de la première variable N (w étant la partie qui dérive de la

deuxième occurrence de la variable 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, ce qui correspond bien aux mots de la forme uv^iwx^iz (dessin de droite). Ces mots doivent donc aussi appartenir au langage engendré par cette grammaire.

9.3 Grammaires et langages contextuels

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).

9.3.1 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 $S \to \varepsilon$ (on peut produire le mot vide depuis la variable de départ) ou de la forme $\alpha X \beta \to \alpha\gamma\beta$, où α, β et γ sont des mots quelconques de $(V \cup \Sigma)^*$, X est une variable de V et $\gamma \neq \varepsilon$ (d'où la nécessité de permettre aussi la règle $S \to \varepsilon$). La nouveauté principale ici (en comparaison des grammaires hors-contexte), est que l'activation 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. Par 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 $\mathtt{a}^n(BC)^n$. 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 ($\mathtt{b}$ et $\mathtt{c}$), à condition qu'ils soient à la bonne place (grâce au contexte !).

Voyons comment produire le mot `aabbcc` (les numéros des règles utilisées sont inscrits à côté de chaque flèche) :

```
S
→2  aSBC
→1  aaBCBC
→3  aaBCZC
→4  aaBWZC
→5  aaBWCC
→6  aaBBCC
→7  aabBCC
→8  aabbCC
→8  aabbcC
→9  aabbcC
→10 aabbcc
```

En terme de machine les reconnaissant, les langages contextuels correspondent exactement aux *automates 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).