Langages formels et automates – cours 9

Langages hors contexte et hiérarchie de Chomsky

Catalin Dima

Au delà des langages réguliers

- Il existe des langages non-réguliers :
  - $L_{anbn} = \{a^n b^n \mid n \in \mathbb{N}\}$ et tous ceux vus en TD.
- On a vu, intuitivement, comment les générer :

$$X = aXb + \varepsilon \quad \text{“génère”} \quad L_{anbn}$$

- Ces sont des langages importants :
  - L_{anbnb} = (sous)-langage des parenthèses.
- **But :** Donner des moyens de construction de langages :
  - Construction algorithmique du langage : énumérer ses mots à l'aide d'un algorithme.
  - Ou identification procédurale : mise à disposition d'une procédure permettant d'identifier les mots appartenant au langage
    - Même si cette procédure peut ne pas se terminer lorsqu'elle ne reconnait pas un mot !

Grammaires

- Généralisation des équations de langages, de type $X = aXb + c$.
- **Grammaire** : $G = (N, T, S_0, P)$ où :
  1. N = les symboles non-terminaux, ou les inconnues de nos équations.
  2. T = les symboles terminaux, ou l'alphabet de nos équations.
  3. S_0 = le symbole de start.
  4. P = les productions, qui sont exactement (des généralisations de) nos équations :

$$P \subseteq (N \cup T)^* N (T \cup N)^* \times (T \cup N)^*$$

- On va écrire une production comme $x \longrightarrow y$,
- On va dire alors que x dérive en/produit y, ou que y dérive de x.
- x est le membre gauche ou le source de la production.
- y est le membre droit ou le résultat de la production.
- Exemple :

$$N = \{X, Y\}, T = \{a, b\}, S_0 = X$$
$$P = \{aXa \longrightarrow abc, XY \longrightarrow aXab, X \longrightarrow aab, YY \longrightarrow abXYa\}$$

Dérivations

- Les productions s'appliquent à certains mots, pour produire de nouveaux mots.
  1. Une production $x \longrightarrow y$ peut s'appliquer à un mot $w \in (N \cup T)^*$, si $w = w_1 x w_2$.
  2. Le résultat de l'application de cette production à w sera $w_1 y w_2$.
- On dénote

$$w_1 x w_2 \Longrightarrow_{x \longrightarrow y} w_1 y w_2$$

et aussi, de manière plus générale,

$$w_1 x w_2 \Longrightarrow w_1 y w_2$$

- Exemples pour
$P = \{aXa \longrightarrow abc, XY \longrightarrow aXab, X \longrightarrow aab, YY \longrightarrow abXYa\}$:

$$a\underline{YY}a \overset{YY \to abXYa}{\Longrightarrow} aab\underline{XY}aa \overset{XY \to aXab}{\Longrightarrow} aab\underline{aXa}baa \overset{aXa \to abc}{\Longrightarrow} aababcbaa$$

- Ce type de chaîne s'apelle dérivation.
- On écrit aussi, de manière plus compacte,

$$aYYa \Longrightarrow^* aababcbaa, \text{ et aussi } aYYa \Longrightarrow^* aabaXabaa$$

Dérivations et langage

- Ce qui nous intéresse c'est les dérivations qui commencent en S_0 et se terminent avec un mot dans T^*.
- On dit alors que le mot est **généré** par la grammaire.
- Le **langage généré** par la grammaire G est l'ensemble de tous les mots générés par la grammaire.
- Et on dénote :

$$L(G) = \{w \in T^* \mid S_0 \longrightarrow w\}$$

- Exemple de mots générés :

$$N = \{X, Y\}, T = \{a, b\}, S_0 = X$$
$$P = \{X \longrightarrow aYYa, aXa \longrightarrow abc, XY \longrightarrow aXab, X \longrightarrow aab, YY \longrightarrow abXYa\}$$
$$X \longrightarrow a\underline{YY}a \Longrightarrow aab\underline{XY}aa \Longrightarrow aab\underline{aXa}baa \Longrightarrow aababcbaa$$

- Le mot *aababcbaa* fait partie du langage généré par cette grammaire.

Dérivations et équations de langages

- On peut regarder les productions comme un système d'équations, qui sert à générer un langage.
  - Si on a plusieurs productions avec le même membre gauche, alors cela ferait une seule équation :

    $$aXa = abc + XYa + \varepsilon$$

    au lieu de

    $$aXa \longrightarrow abc, aXa \longrightarrow XYa, aXa \longrightarrow \varepsilon \in P$$

- Le langage généré est le plus petit point fixe de l'ensemble des productions.
- Parfois un ensemble de productions peut générer un langage vide.

Grammaires “indépendantes de contexte” ou “hors contexte”

- Limitons-nous à la classe des grammaires pour lesquelles toute production a un nonterminal comme membre gauche.
- On n'accepte que des productions de type $X \longrightarrow aXb$ ou $X \longrightarrow abYaXZc$.
- Donc on n'accepte pas des productions $aXa \longrightarrow abc$, ni $YY \longrightarrow abXYa$!
- Ces grammaires peuvent être décrites comme suit :

$$X \longrightarrow aXb \mid abYaXYc \mid acY$$
$$Y \longrightarrow aYYaa \mid \varepsilon \mid abc$$

- C'est une autre notation pour le système :

$$X = aXb + abYaXYc + acY$$
$$Y = aYYaa + \varepsilon + abc$$

Grammaires “indépendantes de contexte” ou “hors contexte”

- Un premier exemple :
$$X \longrightarrow aXb \mid \varepsilon$$
  - On écrit assez souvent aussi $X ::= aXb \mid \varepsilon$.
- On sait bien ce que cette grammaire nous génère, non ?
$$L(G) = \{a^n b^n \mid n \in \mathbb{N}\}$$
- Dérivation :
$$X \overset{X \to aXb}{\Longrightarrow} aXb \overset{X \to aXb}{\Longrightarrow} aaXbb \overset{X \to aXb}{\Longrightarrow} aaaXbbb \overset{X \to \varepsilon}{\Longrightarrow} aaabbb$$
- En général :
$$X \overset{X \to aXb}{\Longrightarrow} aXb \underbrace{\overset{X \to aXb}{\Longrightarrow} \ldots. \overset{X \to aXb}{\Longrightarrow} a^n X b^n}_{n \text{ fois}} \overset{X \to \varepsilon}{\Longrightarrow} aaabbb$$
- Forme sententielle = tout mot sur $(N \cup T)^*$ qui peut être dérivé de S_0 (dans notre cas de X).
  - *aaaaaaXbbbbbb*, X et *aabb* sont des formes sententielles.
  - XX ou *aXab* ou *aXbb* ne le sont pas !

Pourquoi “indépendantes de contexte” ?

- Prenons une production générale $aXa \longrightarrow aYcaa$.
  - On dit que X dérive en Yca sous le contexte $a \cdot a$.
  - Cette production *ne s'applique pas* sur un mot bXa !

$$bXa \stackrel{aXa \to aYcaa}{\not\Rightarrow} \dots$$

  - Pour que X dérive Yca, il faut le contexte $a \cdot a$!
- Prenons maintenant une production “indépendente de contexte” $X \longrightarrow Yca$.
- Cette production peut s'appliquer dans n'importe quel contexte :

$$bXa \stackrel{X \longrightarrow Yca}{\Longrightarrow} bYcaa$$

- Dans une grammaire indépendente de contexte (ou hors contexte, pour faire plus court !), une production s'applique dans n'importe quel contexte.
- On a seulement besoin que le nonterminal à gauche de la production apparaisse dans le mot à dériver.

Pourquoi grammaires ?

- C'est un liguiste qui les a inventées.
- Grammaire usuelle :

$$
\begin{aligned}
\langle \texttt{phrase} \rangle &::= \langle \texttt{proposition} \rangle . \mid \\
&\qquad\quad \langle \texttt{proposition} \rangle . \langle \texttt{phrase} \rangle \mid \ldots \\
\langle \texttt{proposition} \rangle &::= \langle \texttt{sujet} \rangle \langle \texttt{predicat} \rangle \mid \\
&\qquad\quad \langle \texttt{sujet} \rangle \langle \texttt{predicat} \rangle \langle \texttt{complement} \rangle \ldots \\
\langle \texttt{sujet} \rangle &::= \langle \texttt{substantif} \rangle \mid \langle \texttt{pronom} \rangle \ldots \\
\langle \texttt{pronom} \rangle &::= \textit{je} \mid \textit{tu} \mid \ldots
\end{aligned}
$$

- Parfois des règles dépendentes de contexte.

Arbre syntaxique

- Une dérivation qui commence avec un nonterminal dans une grammaire hors contexte peut se représenter graphiquement par l'arbre syntaxique.
- L'arbre est représenté racine en haut.
- Dans la racine, on met le nonterminal qui lance la dérivation.
- Un noeud et tous ses fils représentent une production.
- La frontière de l'arbre = les noeuds sans fils, forment le résultat de la dérivation.

Exemple d'arbre syntaxique

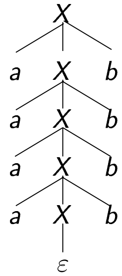
- Prenons notre cas de grammaire simple :

$$X \longrightarrow aXb \mid \varepsilon$$

- Pour générer $a^4 b^4$ on a la suite de dérivations

$$X \Longrightarrow aXb \Longrightarrow aaXbb \Longrightarrow aaaXbbb \Longrightarrow aaaabbbb$$

- L'arbre syntaxique pour cette dérivation :

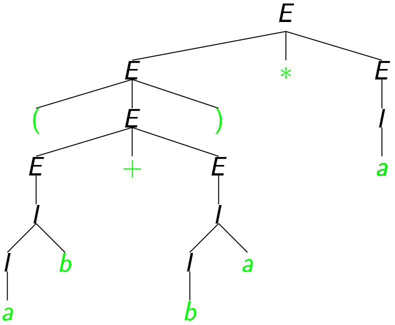


Quelques grammaires

- Grammaire des expressions arithmétiques :

$$E \longrightarrow I \mid E + E \mid E * E \mid (E)$$
$$I \longrightarrow a \mid b \mid Ia \mid Ib$$

  - E est le nonterminal de start.
  - Le nonterminal I génère en effet un *langage régulier* ! Lequel ?
- Exemple d'arbre syntaxique générant $(ab + ba) * a$:



- Faire de même avec $ab * (a + b * ba) + aa$ et $((ab)) + (a)$.

Quelques grammaires et leurs langages

- Exemples vus en TD :

$$L_1 = \{a^n b^{2n} \mid n \in \mathbb{N}\}$$
$$L_2 = \{w1^n \mid w \in \{a,b\}^*, n \in \mathbb{N}, n = \#_a(w)\}$$

- Le langage des parenthèses bien balancées :

$$B \longrightarrow (B) \mid [B] \mid \{B\} \mid BB \mid \varepsilon$$

- Langages de programmation usuels...
- Langages réguliers (eh oui !)
  - Tout langage régulier peut être généré par une grammaire hors contexte.
- Classe des langages **hors contexte** : langages qui sont générés par une grammaire hors contexte.

Algorithmique des grammaires – problématique

- D'abord, les trois problèmes classiques :
  1. Problème de l'*appartenence* : étant donnés une grammaire $G = (N, T, S_0, P)$ et un mot $w \in T^*$, appartient-il au langage de la grammaire, $w \in L(G)$?
  2. Problème de *langage vide* : étant donnée une grammaire G, est-ce que son langage est vide ?
  3. Problème de *langage infini* : étant donnée une grammaire G, est-ce que son langage est infini ?
- Cherche algorithmes !
- Quelques idées ?...

Appartenance

- On pourrait essayer un algorithme de force brute :
  1. Générer toutes les formes sententielles, jusqu'à *une certaine taille*.
  2. Et vérifier si notre mot en fait partie.
- Mais quelle taille ?... car on peut avoir des ε-productions !
- Encore ces ε qui nous embêtent !
- Il faut donc les éliminer !
  - Car si on n'a pas d'ε-productions, on génère les formes sententielles jusqu'à la taille du mot donné !
- Autres opérations pourraient être utiles :
  - Éliminer des nonterminaux inaccessibles ou ceux ne générant que des mots qui contiennent des non-terminaux !
- On a aussi d'algorithmes plus performants pour certains cas particuliers !
  - Voir cours de compilation !

Simplifier un peu une grammaire hors contexte

- Nonterminal inaccessible : ne sera présent dans aucune forme sententielle (ou dans aucun arbre syntaxique).
- Algorithme simple d'élimination de nonterminaux inaccessibles :
  1. On construit itérativement l'ensemble de nonterminaux accessibles N_{acc}.
  2. S_0 est toujours accessible, donc on le met dans N_{acc}.
  3. Pour chaque $X \in N_{acc}$, on prend chaque production $X \longrightarrow w \in P$ et on rajoute dans N_{acc} les nonterminaux dans w.
  4. On s'arrête lorsque N_{acc} ne grossit plus.
- Après avoir fait cela, on élimine aussi toutes les productions qui utilisent des nonterminaux dans $N \setminus N_{acc}$.

Éliminer autres nonterminaux inutiles

- Nonterminal non-génératif : toutes les dérivations de X contiennent des nonterminaux.
- Exemple très simple : considérons la grammaire ayant que la production

$$X \longrightarrow aXb$$

  - Langage vide ! on ne peut générer aucun mot dans $\{a, b\}$ sans nonterminal !
- Construction de l'ensemble des nonterminaux génératifs N_{gen} :
  1. Tout nonterminal X qui possède une production $X \longrightarrow w$ avec $w \in T^*$ est mis dans N_{gen}.
  2. À chaque pas, si on trouve un $X \in N \setminus N_{gen}$ mais pour lequel on a une production $X \longrightarrow z$ pour laquelle tous les nonterminaux de z sont dans N_{gen}, on rajoute X dans N_{gen}.
  3. Encore une fois, on s'arrête lorsque N_{gen} ne grossit plus.
  4. Après cela, on élimine toutes les productions impliquant les nonterminaux qui ne sont pas dans N_{gen}.

Éliminer les nonterminaux

- Il faut éliminer les nonterminaux non-génératifs avant d'éliminer les inaccessibles !
- Exemple :

$$
\begin{aligned}
S &\longrightarrow AB \mid CA \\
A &\longrightarrow a \\
B &\longrightarrow BC \mid AB \\
C &\longrightarrow aB \mid a
\end{aligned}
$$

Éliminer les ε-productions

- On construit itérativement l'ensemble des nonterminaux qui peuvent produire ε, N_ε.
  1. On initialise N_ε avec tous les terminaux X pour lesquels il existe une production $X \longrightarrow \varepsilon$.
  2. On rajoute à N_ε un nonterminal X lorsque $X \longrightarrow z \in P$ et z se compose que des nonterminaux déjà dans N_ε.
  3. On s'arrête lorsque N_ε ne grossit plus.
- Après cela, on élimine toutes les ε-productions.
- Mais il faut aussi modifier certaines autres productions !
  - Si on a $X \longrightarrow xYy$ avec $Y \in N_\varepsilon$ on va aussi rajouter la production $X \longrightarrow xy$ dans P !
  - Car il faut aussi prendre en compte le fait que Y produit ε !
- Un seul problème reste : qu'est-ce qu'on fait lorsque $S_0 \in N_\varepsilon$?...
- Éliminer toutes les ε-productions changerait le langage !
- Alors on rajoute un nouveau nonterminal S_0' avec deux productions :

$$S_0' \longrightarrow S_0 \mid \varepsilon$$

- C'est le seul à qui on permettra d'avoir des ε-productions.
- Il faut prouver que le langage de la nouvelle grammaire G' reste le même que celui de la grammaire donnée G !

Encore des éliminations

- Il y a encore des productions qui nous gênent un peu dans notre algorithme de l'appartenence :
  - Renommages, c.à.d. productions de type $X \longrightarrow Y$, $X, Y \in N$.
  - Elles nous gênent car leur application sur une forme sententielle ne fait pas grossir celle-ci.
  - Et donc ne nous approche pas de notre but, construire toujours des formes sententielles plus grandes.
- Il faut donc les éliminer !
  - Une première idée : si on a $X \longrightarrow Y$ et $Y \longrightarrow w$, alors on rajoute $X \longrightarrow w$.
  - Sauf qu'on peut avoir des cycles, donc on ne sait pas quand éliminer $X \longrightarrow Y$...
- On va donc procéder en deux étapes, comme pour les ε-productions

Éliminer les renommages

- On va d'abord construire les renommages "inductives", c'est à dire les séquences de dérivations $X \Longrightarrow Y$, $X, Y \in N$.
- Pour chaque nonterminal X, on construit l'ensemble des renommages de X :

$$R_X = \{Y \in N \mid X \Longrightarrow Y \text{ dans } G\}$$

- Ces ensembles se construisent itérativement aussi.
- Puis on élimine d'un seul coup tous les renommages, mais en rajoutant des productions supplémentaires :
  - Pour tout $Y \in R_X$ et tout $Y \longrightarrow z \in P$ rajouter $X \longrightarrow z$ dans le nouvel ensemble de productions.
- Exemple (avec élimination d'ε-productions) :

$$\begin{aligned} S &\longrightarrow ASB \mid \varepsilon \\ A &\longrightarrow aAS \mid a \longrightarrow \varepsilon \\ B &\longrightarrow SbS \mid A \mid bb \end{aligned}$$

Langage vide et infini

- Langage vide : S_0 est non-génératif.
- Langage infini : on doit avoir une "boucle" entre non-terminaux.
  1. On élimine tous les nonterminaux inutiles.
  2. On élimine les ε-productions et les renommages.
  3. Et maintenant on construit, pour tout nonterminal $X \in N$, les nonterminaux qui sont atteignables à partir de X,

$$Att(X) = \left\{ Y \mid X \Longrightarrow^* xYz \right\}$$

  4. Le langage est infini si et seulement si $X \in Att(X)$ pour un $X \in N$.

Ambiguité

- On peut avoir plusieurs arbres syntaxiques qui génèrent le même mot.
- Situation indésirable, surtout en lanages de programmation !
  - On ne sait pas comment traduire le programme !
- On parle alors de grammaire ambiguë.
- Il faut donc éviter d'utiliser des grammaires ambigues, éventuellement en les transformant en grammaires non-ambigues.
- Notre grammaire des expressions arithmétiques est ambiguë !

$$E \longrightarrow I \mid E + E \mid E * E \mid (E)$$
$$I \longrightarrow a \mid b \mid Ia \mid Ib$$

  - On a deux arbres syntaxiques pour $a + b + ab$!
- Autre exemple :

$$S \longrightarrow (S) \mid SS \mid \varepsilon$$

- Les ε-productions ne sont pas une source d'ambiguité !

Ambiguité

- Pour le langage hors contexte des expressions arithmétiques on peut donner une grammaire non-ambiguë :

$$
\begin{aligned}
E &\longrightarrow T \mid T + E \\
T &\longrightarrow F \mid F * T \\
F &\longrightarrow I \mid (E) \\
I &\longrightarrow a \mid b \mid Ia \mid Ib
\end{aligned}
$$

- Mais il y a des langages hors contexte pour lesquels on ne peut pas donner de grammaire non-ambiguë !
  - Exemple :

$$L_{ambg} = \{a^n b^n c^m d^m \mid m, n \in \mathbb{N}\} \cup \{a^n b^m c^m d^n \mid m, n \in \mathbb{N}\}$$

- On dit alors que L_{ambg} est un langage à ambiguité inhérente.

Et les autres cas de grammaires ?

- Grammaire dépendant de contexte ou sous contexte, ou encore de type 1 :
  - Toute production doit respecter le format $xXy \longrightarrow xwy$, ou $X \in N$ (et $x, y, w \in (N \cup T)^*$ comme d'habitude).
- Grammaire de type 0 : sans contrainte sur les productions.
- Les grammaires hors contexte portent aussi le nom de grammaires de type 2.
- Et il y a aussi les grammaires de type 3 ou régulières :
  - Productions de type $X \longrightarrow wY$ où $w \in T^*$ et $X, Y \in N$.
  - Ces sont des grammaires régulières à gauche.
  - On peut avoir aussi des grammaires régulières à droite.
- Chaque classe de grammaires génère une classe de langages.
  - On a donc les langages réguliers $\mathcal{R}eg$, hors contexte CF, sous contexte CS et de type 0 $\mathcal{L}_0$.
  - Et les inclusions $\mathcal{R}eg \subsetneq CF \subsetneq CS \subsetneq \mathcal{L}_0$.
- Inclusions strictes !
- Déjà on le sait pour $\mathcal{R}eg \subsetneq CF$! (c'est $a^n b^n$ qui le prouve).

Grammaires sous contexte

- On a vu que $L_{anbn} = \{a^n b^n \mid n \in \mathbb{N}\}$ est hors contexte.
- Mais qu'en est-il de

$$L_{anbncn} = \{a^n b^n c^n \mid n \in \mathbb{N}\}?$$

- Grammaire pour ce langage :

$$\begin{aligned} S &\longrightarrow aSBC \mid \varepsilon \\ CB &\longrightarrow BC \\ aB &\longrightarrow ab, bB \longrightarrow bb, bC \longrightarrow bc, cC \longrightarrow cc \end{aligned}$$

- On peut donner une grammaire sous contexte aussi !
- Comment prouver que ce langage n'est pas un langage hors contexte ?
  - Il nous faut un lemme similaire au lemme de l'étoile !