0.6 Transformation de grammaires

Un peu comme avec les automates, on va s'intéresser ici à différentes façon de transformer une grammaire en une autre grammaire (de même pouvoir génératif faible), lorsque la grammaire initiale présente des propriétés non désirables. En fait, dans ce cours niveau L3, on va simplement s'intéresser au nettoyage des grammaire.

0.6.1 Grammaires propres

Il s'agit de manipulations liées à la présence potentielle, dans une grammaire, de règles redondantes ou inutiles.

Une grammaire est ***propre*** si elle est :
- ε-libre,
- dépourvue de symboles inutiles,
- sans cycle

En deux mots, ε-libre signifie qu'il n'y a pas de production donnant ε, par symbole inutile on entend à la fois ceux qui n'ont pas de contribution et ceux qui sont inaccessibles. Enfin, les cycles impliquent des productions singulières qui peuvent engendrer des boucles inutiles dans une dérivation.

Définitions plus rigoureuses

Définition Une grammaire algébrique est dite *ε-libre* si
- P n'a pas d'ε-production, ou
- P a exactement une ε-production $S \longrightarrow \varepsilon$, et S est inaccessible (*i.e.* S est l'axiome, *et* aussi S n'apparaît pas dans un membre droit de règle).

Définition Un symbole non-terminal X sera dit *inutile* si X est sans contribution, ou inaccessible. Un symbole non-terminal X est dit *sans contribution* s'il n'existe pas de dérivation $X \xrightarrow{*} u$, $u \in X^*$. Un symbole non-terminal X est dit *inaccessible* s'il n'existe pas de dérivation $S \xrightarrow{*} wXy$ $(w, y \in (X \cup V)^*)$.

Définition Une grammaire sans cycle (*cycle-free*) ne contient aucune dérivation $A \xrightarrow{+} A$ pour tout $A \in V$. Pour débarrasser une grammaire des cycles potentiels, on supprime les **productions singulières**.

Grammaires ε-libres

Exemple $S \longrightarrow aSbS \mid bSaS \mid \varepsilon$ devient (S' axiome) : $S \longrightarrow aSbS \mid abS \mid aSb \mid ab$
$S \longrightarrow bSaS \mid baS \mid bSa \mid ba$
$S' \longrightarrow S \mid \varepsilon$

Algorithme

1. On construit $\mathcal{A} = \{A \in V / A \xrightarrow{+} \varepsilon\}$
2. Pour toute règle $B \longrightarrow \alpha_0 A_0 \alpha_1 A_1 ... \alpha_n$ où un nombre non nul des A_i appartiennent à l'ensemble $\mathcal{A}$, on remplace la règle par toutes les règles $B \longrightarrow \sigma(\alpha_0 A_0 \alpha_1 A_1 ... \alpha_n)$ où $\sigma(m)$ est l'ensemble de toutes les combinaisons où les A_i annulables sont remplacés

ou non par ε (en omettant un cas : si le corps de la règle serait composé uniquement d'annulables, alors on exclut $b \longrightarrow \varepsilon$)

3. Si $S \in \mathcal{A}$, ajouter S' dans V, et $S' \longrightarrow S \mid \varepsilon$ dans P, et remplacer l'axiome S par S'.

Suppression des inutiles

Algorithme de recherche des X sans contribution On construit récursivement les ensembles $N_0, N_1, ..., N_k = N_{k+1}$ contenant les symboles non-terminaux qui contribuent.

$i = 0$
$N_0 = \emptyset$
repeat
  $i = i + 1$
  $N_i = N_{i-1} \cup \{A \in V \,/\, A \to \alpha \text{ et } \alpha \in (N_{i-1} \cup X)^*\}$
jusqu'à $N_i = N_{i-1}$
$N_e = N_i$

Si $S \in N_e$ alors $L(S) \neq \emptyset$.

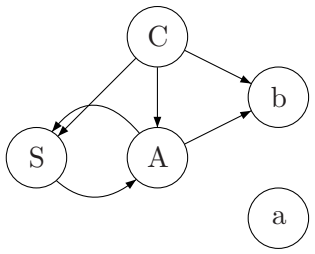
Cet algorithme fournit la liste de symboles ayant une contribution, on peut donc débarrasser la grammaire de tous les symboles sans contribution (et les règles les concernant).

Exemple :

$S \longrightarrow A \mid B$	$N_1 = \{A, C\}$	Version simplifiée :
$A \longrightarrow aB \mid bS \mid b$	$N_2 = \{A, C, S\}$	$S \longrightarrow A$
$B \longrightarrow AB \mid Ba$	$N_3 = \{A, C, S\}$	$A \longrightarrow bS \mid b$
$C \longrightarrow AS \mid b$	$L_G(B) = \emptyset$, donc B ss cont.	$C \longrightarrow AS \mid b$

Algorithme de recherche des inaccessibles L'algorithme exploite le graphe orienté obtenu à partir de la grammaire de la façon suivante : les sommets sont tous les symboles de $X \cup V$; les arcs sont définis ainsi : il y a un arc de s_1 à s_2 si il existe une règle de la forme $s_1 \longrightarrow m_1 s_2 m_2$ dans P, avec $m_1, m_2 \in (X \cup V)^*$.

Exemple : (grammaire précédente, non simplifiée)



[Noter que si on s'intéresse ici seulement aux non terminaux, la notion d'inaccessible peut aussi s'appliquer aux symboles terminaux.] Un parcours de ce graphe (en largeur, avec marquage) à partir de l'axiome marque tous les symboles (terminaux ou non terminaux) qui sont accessibles. Dans l'exemple, seul C n'est pas marqué.

Comme pour l'algo précédent, il ne reste plus qu'à enlever les terminaux non marqués.

D'où la forme finale de la grammaire (en partant de la version simplifiée précédente) :

$S \longrightarrow A$
$A \longrightarrow bS \mid b$

Une autre façon de voir les choses (cf. Hopcroft/Ullman) par algorithme de point fixe :

Base : S est accessible

Induction : si A est accessible, alors pour toutes les productions avec A dans le membre gauche, tous les symboles du membre droit sont accessibles

On s'arrête au point fixe.

Grammaire sans cycles Les cycles sont liés à la présence de Productions singulières :
$S \longrightarrow A \mid a$
$A \longrightarrow B$
$B \longrightarrow C$
$C \longrightarrow A$

Attention, ce n'est pas parce qu'on a des productions singulières qu'on a nécessairement un cycle : $G_3 = E \to E + T \mid T, T \to T \times F \mid F, F \to (E) \mid a$

Productions singulières

Définition Règles de la forme $A \longrightarrow B$, avec $A, B \in V$. Pour enlever les productions singulières, on part d'une grammaire ε-libre.

Algorithme On commence par trouver toutes les paires singulières de la grammaire. Une paire singulière est un couple (A, B) tel que $A \xrightarrow{*} B$ (en n'utilisant que des productions singulières).

Note : il est possible d'avoir $A \xrightarrow{*} B$ sans utiliser de productions singulières : $A \longrightarrow BC; C \longrightarrow \varepsilon$. Si on a nettoyé la grammaire des ε-productions avant le cas ne se produit pas.

Pour trouver toutes les paires singulières on procède de la façon suivante :

Base : (A, A) est une paire singulière pour tout non terminal A ($A \xrightarrow{*} A$ en 0 étapes)

Induction : on suppose que (A, B) est une paire singulière, si $B \longrightarrow C$ est une règle de production alors (A, C) est une paire singulière

Pour éliminer les productions singulières on construit une grammaire $\langle V, T, S, P_1 \rangle$ à partir de $\langle V, T, S, P \rangle$ la grammaire de base

1 : Trouver toutes les paires singulières de la grammaire

2 : Pour toute paire singulière (A, B), on ajoute à P_1 l'ensemble des règles de la forme $A \longrightarrow \alpha$ où α est une production non-singulière de B (i.e. $\alpha \in (X \cup V)^*$ et il existe une règle de la forme $B \longrightarrow \alpha$)
Lorsque $A = B$ on ajoute donc toutes les productions non-singulières de A à P_1.

Les productions singulières ne sont pas très gênantes, mais elles sont d'une certaine façon inutiles, et ont le mauvais goût de permettre l'apparition de cycles.

Application avec G_3 la grammaire des expressions arithmétiques.

0.6.2 Résumé

Le nettoyage d'une grammaire se fait donc dans l'ordre suivant :

1. Élimination des ε-productions
2. Élimination des productions singulières
3. Élimination des symboles inutiles

Pourquoi cet ordre : 2 n'introduit pas d'ε-productions, donc conserve les résultats de 1. 3 ne fait que supprimer des règles, donc n'introduit ni productions singulières, ni ε-productions.