Notes du module de grammaires lexicalisées

Hugo Mougard

17 septembre 2013

0.1 Grammaires lexicalisées

Une grammaire lexicalisée est une grammaire dont toutes les règles contiennent au moins un terminal.

Forme normale de Greibach

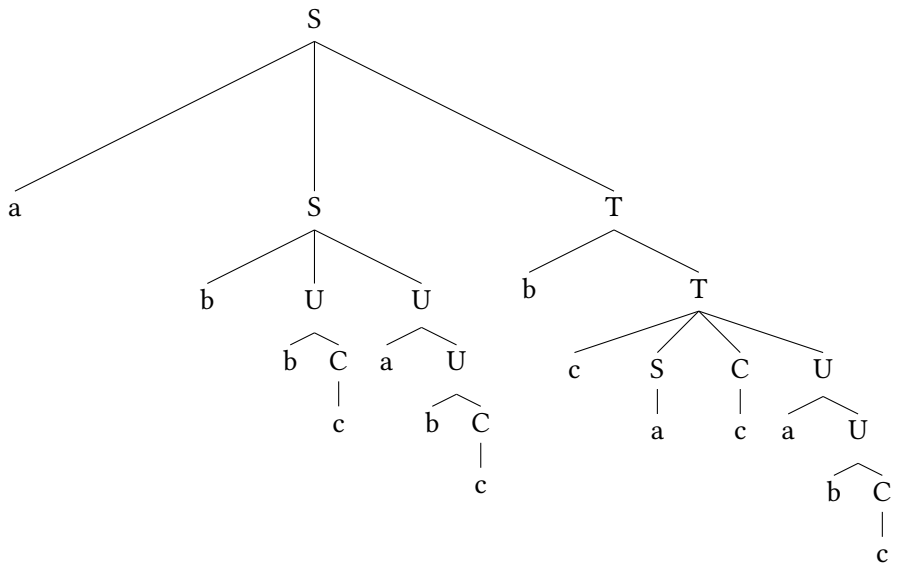
Concerne les langages propres (sans ε) dont les grammaires (propres elles aussi, ou réduites) sont sous la forme

$$V \longrightarrow XV^*$$

où X est terminal et V non-terminal.
Par exemple, voici une grammaire lexicale :

$S \longrightarrow aST \mid bUU \mid a$
$T \longrightarrow bT \mid cSCU \mid cTU$
$U \longrightarrow aU \mid bC$
$C \longrightarrow c$

Et l'arbre de parsing de la phrase abbcabcbcacabc en utilisant cette grammaire :



On peut aussi « inverser » la grammaire pour la centrer sur les terminaux :

a ⟶ S
aST ⟶ S
aU ⟶ U
bUU ⟶ S
bT ⟶ T
bC ⟶ U
cSCU ⟶ T
cTU ⟶ T
c ⟶ C

Traduction en automate à piles :

En utilisant la grammaire inversée ci-dessus, on peut produire le tableau ci-dessous :

	S	T	U	C
a	ε		U	
	ST			
b	UU	T	C	
c	SCU			ε
	TU			

Le tableau se lit, par exemple pour la case (a, S) : on peut dépiler S si on lit a et qu'on empile ε ou T puis S.

Exemple d'exécution :

bande : abb, pile S
⟶ lit a
⟶ bande : bb, dépile S, empile ST, pile ST
⟶ lit b
⟶ bande : b, dépile S, empile UU, pile UUT
⟶ lit b
⟶ bande : #, dépile U, empile C, pile CUT

Transformation en forme normale de Greibach :

Tout d'abord, il faut noter quels sont les problèmes :
— les règles produisant ε ne sont pas tolérées (on vise une grammaire propre)
— les règles récursives à gauche ne sont pas tolérées

Voici maintenant l'algorithme pour transformer les règles récursives à gauche en règles récursives à droite :

Pour tout V_i non terminal, on introduit V_i' puis, on transforme tout règle de la forme :

$$V_k \longrightarrow V_k M_1 \mid \ldots \mid V_k M_n \mid W_1 \mid \ldots \mid W_p$$

en

$$V_k \longrightarrow W_1 V_k' \mid \ldots \mid W_p V_k' \mid W_1 \mid \ldots \mid W_p$$
$$V_k' \longrightarrow M_1 V_k' \mid \ldots \mid M_n V_k' \mid M_1 \mid \ldots \mid M_n$$

puis

$$V_j \longrightarrow V_k M$$
$$V_j \longrightarrow W_1 V_k' M \mid \ldots \mid W_p V_k' M \mid W_{1M} \mid \ldots \mid W_{pM}$$

devient

$$V_j \longrightarrow W_1 V_k' m \mid \ldots \mid W_p V_k' M \mid W_1 M \mid \ldots \mid W_p M$$

Exemple :

$$A \longrightarrow BC$$
$$B \longrightarrow AB \mid a$$
$$C \longrightarrow AC \mid b$$

devient

A ⟶ BC
B ⟶ BCB | a
C ⟶ BCC | b

puis

B ⟶ aB' | a
B' ⟶ CBB' | CB
A ⟶ aB'C | aC
C ⟶ aB'CC | aCC | b

puis

B ⟶ aB' | a
B' ⟶ aB'CCBB' | aCCBB' | bBB' | AB'CCB | aCCB | bB
A ⟶ aB'C | aC
C ⟶ aB'CC | aCC | b

Exercice

1. Mettre sous forme lexicale :

   A ⟶ AaB | BA | b
   B ⟶ Bd | BAa | aA | c

2. Mettre sous forme lexicale :

   S ⟶ AA | a
   A ⟶ SS | b

3. Tout langage algébrique propre peut s'écrire avec une grammaire sous forme normale de Greibach double :

   V ⟶ X
   V ⟶ XV*X

   Apporter un argument.

0.2 Head-driven Phrase Structure Grammar