Forme normale de Greibach

Arthur MILCHIOR

19 décembre 2008

1 Definitions

Pour parler des formes normales de Greibach, il est nécessaire de connaitre les langages algèbriques, les grammaires hors contextes et les automates à piles. Je présume connu le chapitre "langage algèbrique" de [2]

Commençons par définir ce que sont la forme normale de Greibach et ses variantes. Bien qu'on puisse montrer qu'elles reconnaissent toutes les langages algébriques privés du mot vide, donc qu'elles sont toutes équivalentes aux grammaires hors contextes quelconques — ce qui ne sera pas fait systèmatiquement ici — elles ont un intéret, puisque des algorithmes peuvent s'éxécuter beaucoup plus efficacement si l'on sait que la grammaires a une certaine forme.

Définition 1. (Forme normale de Greibach) Une grammaire est dite en forme normale de Greibach si chacun des membres droits de ses règles commence par un terminal.

Définition 2. (Forme normale de quasi-greibach) Une grammaire est dites en forme quasi-Greibach si les membres droit de ces règles sont ε ou un mot commençant par un terminal.

Certaines définitions des formes normales de (quasi-)Greibach acceptent que la variable initiale puisse se dériver en le mot vide. Cela permet d'accepter les langages ayant le mot vide. Dans ce travail de rédaction je ne parlerai que de langages n'acceptant pas le mot vide.

Il est évident, d'après leurs définitions, que les formes normales de Greibach sont des grammaires propres.

Proposition 3. *Sur une grammaire propre toute dérivation ne réduit ni le nombre de terminaux ni le nombre de lettres de $A+V$, et un de ces deux nombre est strictement augmenté.*

Démonstration. Puisque la grammaire est normale, alors il suffit de remarquer que chaque variable ne peut pas se dériver en le mot vide, et que les terminaux ne disparraissent pas. La première partie de la phrase est donc vraie. En supposant que le nombre de terminaux n'augmente pas, alors le membre droit de la dérivation ne contient que des variables, et en contient au moins deux, donc la deuxième partie de la proposition est vraie. □

Cela a un double intérets pour reconnaitre un mot de taille n, d'une part le nombre de dérivation est au plus 2n, d'autre part si lors d'une dérivation on obtient un mot de taille supérieure à n c'est que l'on a choisi une mauvaise branche.

Proposition 4. *Soit une grammaire en forme normale de Greibach, et m un mot de la forme (A+V)*, alors toute dérivation incrémente le nombre de terminaux.*

Démonstration. Encore une fois, c'est évident, il suffit de remarquer que une dérivation transformera une variable en un mot de AV*, donc que le nombre de terminaux est incrémenté. □

Cette propriétée nous apprend donc qu'il faut exactement n dérivations pour pouvoir reconnaitre un mot composé de n terminaux.

Si l'on appelle G(v,t) l'ensembles des membres droit de la forme tV* des règles dont le membre gauche est v, alors on a un algorithme évident, qui termine toujours, et qui est linéaire de manière non déterministe.

Algorithm 1 reconnait(m,v)

```
if m == ε et v == ε then
  rendre vrai
if m! = ε ou v! = ε then
  for all r ∈ G(head(m), head(v)) do
    if reconnait(tail(m),tail(r)tail(v) then
      rendre vrai
rendre faux
```

Voyons maintenant des variantes de la forme normale de Greibach.

Définition 5. (Forme normale de Greibach quadratique et cubique) Une grammaire en forme normale de Greibach est dite quadratique (resp. cubique) si chaque règle ne comporte au plus que 2 (resp. 3) variables.

Définition 6. (Forme normale de Greibach double) Une grammaire est en forme normale de Greibach double si chaque règle commence et finit par un terminal. Les membres droits sont donc de la forme A ou AV*A.

Il est bien entendu possible de combiner les deux variations pour parler de forme normale de Greibach double quadratique ou cubique.

2 Construction

Dans cette partie, à partir d'une grammaire quelconque, nous allons construire une grammaire en forme normale de Greibach quadratique reconnaissant le même langage. Je suivrai pour cela [1].

Il faut préalablement introduire de nouvelles définitions et quelques propriétées triviales.

2.1 Pseudo-automate à pile

Définition 7. (pseudo-automate à pile) On appelle pseudo-automate à pile un automate à pile qui peut lire un nombre arbitraire de symboles de pile pour effectuer une transition. La pile initiale peut être un mot de taille quelconque.

On appelle une ε-transition toute transition ne lisant rien sur la bande.

Une transition est croissante(resp. décroissante) si la taille de la pile croit(resp. décroit) strictement avec cette transition.

Proposition 8. *Les pseudos automates à piles sont équivalents aux automates à piles.*

Démonstration. Les automates à piles sont des pseudos-automates à pile. Il ne reste donc qu'à montrer que tout langage reconnu par un pseudo-automate à pile peut aussi être reconnu par un automate à pile, on va donc donner un algorithme permettant de passer d'un pseudo-automate à pile à un automate à pile.

Lemme 9. *La transformation suivante prend un pseudo-automate à pile et rend un automate à pile reconnaissant le même langage.*

L'alphabet de A est celui de A', l'alphabet de bande de A est celui de A' augmenté d'un nouveau symbole de début de bande #.

L'état initial est q_0 et il y a les transitions $q_0, \varepsilon, \# \to q, m$ où m est le mot de début de pile de A, pour tout état initiale q de A.

Si A' accepte par pile vide, alors il y a la transition $q, \varepsilon, \# \to q, \varepsilon$ pour tout état q.

Si A' accepte par état, alors les mêmes états dans A seront acceptant.

Pour se débarasser des ε-transition $q, l, \varepsilon \to q', m$ on rajoute $q, l, v \to q', vm$ pour chaque variable de pile v de A.

Si la pile pouvait être vide, on aurait eu un problème, c'est pour cette raison que l'on a rajouté #.

Pour pouvoir ne plus avoir de transition lisant une liste de symboles $s_1 s_2 ... s_n$, il suffit de rajouter pour chaque transition $q, l, s_1 s_2 ... s_n \to q', m$ une liste de nouveaux états $(q_i)_{i \in [[2,n]]}$, et les transitions $q_i, \varepsilon, s_i \to q_{i+1}, mi \in [[1, n-1]]$ et $q_n, l, s_n \to q', m$, où l'on confond q_1 et q.

Ces transitions ne font que lire ε sur la bande, et dépilent le $i^{ème}$ symbole de bande de la transition initiale. A la lecture du dernier symbole de la liste, on empilera ce que la transition empilait, et on passera dans l'état dans lequel la transition initiale fait passer.

Les transitions de A' lisant exactement un symbole de bande sont recopiées tel quels dans A.

Démonstration. Il suffit de se convaincre qu'il ne s'agit que d'une traduction, et si le nombre de transition change, puisque depuis q_i on ne peut passer que en q_{i+1} alors les deux automates sont équivalents. □

□

Proposition 10. *Si le pseudo-automate à pile n'empilait jamais plus d'une lettre, alors l'automate à pile produit n'en empile jamais plus de deux.*

Si les ε-transitions sont décroissantes dans A', alors les transitions de A venant de ces ε-transitions le sont toujours dans l'automate à pile obtenu.

Démonstration. La preuve de ces deux propositions est imédiate en regardant la construction. □

2.2 Inversion

Définition 11. (Inverse) Pour tout mot $w = a_1a_2....a_{n-1}a_n$, on note $\tilde{w}$ l'inverse de w, c'est à dire le mot $a_na_{n-1}....a_2a_1$.

Pour toute grammaires G, on note $\tilde{G}$ la grammaire ou chaque membre droit w est remplacé par $\tilde{w}$.

Pour tous langages L, on note $\tilde{L}$ le langage qui contient les inverses des mots de L.

Pour tout automate à pile A, on note $\tilde{A}$' le pseudo-automate à pile obtenu en changeant les règles $e_1, l, x \to e_2, y$ par $e_1, l, y \to e_2, x$. Si A accepte par pile vide et possède # comme symbole de début de pile, alors $\tilde{A}$' a un état e_f d'où ne part aucune transition, et les transitions $e, \varepsilon, \# \to e_f, \varepsilon$ pour chaque état e différent de e_f. Si l'automate accepte par états acceptants, alors les états de départ et d'acceptions sont inversés.

On note $\tilde{A}$ l'automate à pile obtenu à partir de $\tilde{A}$'.

Proposition 12. *$\tilde{L}_G(v) = L_{\tilde{G}}(v)$, pour toute variable v. Autrement dit, l'inverse d'une grammaire G génère le langage inverse de celui généré par la grammaire G.*

Le langage reconnu par $\tilde{A}$' et $\tilde{A}$ est l'inverse du langage reconnnu par A.

Ces propositions sont relativement évidente à voir, intuitivement pour les automates, on tente simplement de lire le mot à l'envers, en partant d'une machine dans l'état dans lequel on devrait être à la fin, et en fonctionnant dans le sens inverse de l'automate inital. Une preuve formelle serait plutôt longue et sans intéret, elle ne sera pas faite ici.

2.3 Construction

La construction d'une grammaire en forme normale de greibach quadratique se fait suivant l'algorithme 2 et nous allons maintenant montrer que cet algorithme construit bien une grammaire en forme normale de greibach quadratique qui engendre le même langage, sauf peut être ε.

Démonstration. La construction de $\tilde{G}$ ne lui apporte aucune propriétée.

La deuxième étape rend cette grammaire propre.

Puisque à la troisième étape, la grammaire est propre alors il n'y a que deux types de règles :

Algorithm 2 greibach-quadratique(G,axiome)

Construire $\tilde{G}$
Rendre $\tilde{G}$ propre
Construire l'automate à pile A reconnaissant le langage généré par G en partant de l'axiome, en utilisant la méthode du cours
Construire l'automate à pile $\tilde{A}$, inverse de A.
Constuire la grammaire G' à partir de $\tilde{A}$
Rendre G' propre.

$(s, x, x) \to (s, \varepsilon)$, qui est décroissante, et $(s, \varepsilon, z) \to (s, z_1 z_2 ... z_n)$, si z_i est la $i^{ème}$ lettre du mot. Les ε-transitions ne sont donc pas décroissantes, puisque il n'y a pas de règle ayant le mot vide à droite.

En quatrième étape, en inversant l'automate à pile, les ε-transitions sont décroissantes par construction. De plus, dans le pseudo-automate à pile construit, on n'empile jamais plus d'un symbole à la fois, donc dans l'automate on n'empile jamais plus de deux symboles.

En cinquième étape, avec la méthode des triplets de Ginsburg, on obtient une grammaire de forme quasi-greibach, qui est quadratique, de par le fait qu'on n'empile jamais plus de deux symboles.

Enfin, puisque l'algorithme rendant une grammaire propre, appliqué à une grammaire en forme Normale de greibach, rend une grammaire propre, alors la grammaire obtenue est bien en forme normale de greibach quadratique. □

En dehors du coté étonnant de cette construction, il n'est pas certain qu'elle soit intéressante, puisque l'algorithme appliqué à une grammaire en forme normale de greibach quadratique rend une grammaire plus grosse. Cela peut se voir par exemple au moment où l'on inverse l'automate à pile $\tilde{A}$, puisque on passe par un pseudo-automate à pile, et que la transformation rajoute le symbole de pile # et une transition initiale en plus des transitions déjà existante.

Références

[1] J.M. Autebert, L. Boasson, and J.Gabarro. Context-free grammars in greibach normal forms. 1984.

[2] O. Carton. *Langages formels calculabilité et complexité.* vuibert, 2008.