Introduction aux langages algébriques et à l'analyse syntaxique

Philippe Durand

novembre 2001

1 Introduction

Les langages algébriques génèneralisent les langages réguliers . Alors qu'il suffit de simples automates pour reconnaitre les expressions rationnelles on doit utiliser des machines plus compliquées pour reconnaitre un langage algébriques. En fait on peut montrer qu'il suffit d'ajouter un peu de mémoire (une pile) a un automate pour identifier un langage algébrique.

2 le lemme de l'étoile

Le langage le plus simple non reconnaissable par A.F est un sous ensemble du langage bien parenthèsé dont tous les mots s'ecrivent sous la forme $a^n b^n$.

Théoreme

Un langage L est regulier si il existe deux entiers n et $m \geq 1$ tel que l'on ait:
$uv^k w \in L$ avec $k \geq n \Rightarrow uv^k(v^m)^* w \subseteq L$

Preuve

En effet si L était reconnaissable il existerait un automate fini à n états le reconnaissant:

$q_0 \mapsto q_1 \mapsto q_2 \mapsto ...q_p \mapsto F$:
et :
$q_0.u = q_1$, $q_1.v^k = q_p$, $q_p.w = F$ avec F état terminal.
Il suffit alors de choisir $k > n$ alors on passe obligatoirement deux fois par le même état: $k = k_1 + k_2 + k_3$ avec:
$q_1.v^{k_1} = q_i$, $q_i.v^{k_2} = q_i$, $q_i.v^{k_3} = q_p$ On en déduit le resultat.•

Exercices

Montrer que les langages suivants ne sont pas reconnaissables par automates finis:

$L_1 = \{a^n b^n, n \in N\}$ avec $\Sigma = \{a, b\}$

$L_2 = \{a^n b^{n+1}, n \in N\}$ avec $\Sigma = \{a, b\}$

$L_3 = \{a^r, r : premier\}$ avec $\Sigma = \{a\}$

$L_4 = \{w \in \Sigma^*$ et w palindrome$\}$ avec $\Sigma = \{a, b\}$

3 Grammaires , hierarchie de Chomsky.

Tous les langages peuvent être décrit par des grammaires plus ou moins complexes. Les langages réguliers entrent dans ce cadre: on peut associer à tout automate fini une grammaire linéaire droite ou gauche. Il suffira de remplacer la transition $X.a = Y$ par la règle de grammaire: $X \mapsto a.Y$. On vera à la fin de cet exposé que l'on peut faire presque la même chose avec les langages algébriques en remplacant grammaire linéaire par grammaire de Greibach et automate par automate à pile...

3.1 Grammaires, définitions, exemples.

Une grammaire est un quadruplet $< N, T, S, R >$ avec:

N: ensemble fini d'éléments non terminaux.
T: ensemble fini d"eléments terminaux: l'alphabet terminal.
S: l'axiome.
P: un ensemble fini de règles de productions, P est une partie finie de:
$(N + T)^* \times (N + T)^*$, et la première projection contient au moins un non terminal: $|pr_1|_N = 1$
un élément de P (u, v) est d'usage noté $u \to v$

3.1.1 exemples

1. Prenons $N = \{S\}$, $T = \{a, b\}$, $P : S \mapsto aSb + \varepsilon$.
2. puis, $N = \{E, T, F\}$, $T = \{a, \oplus, \odot, (,)\}$, $E : axiome$ et P:
   $E \mapsto E \oplus T + T$
   $T \mapsto T \odot F + F$
   $F \mapsto (E) + a$

La deuxième grammaire engendre l'arithmétique sur la lettre a

3.1.2 Langage reconnu par une grammaire

Un mot est reconnu par une grammaire G si, partant de l'axiome on aboutit à w en n pas de d{erivation, un pas de dérivation étant l'usage d'une seule règle de production. Par le thèorème de Tarski on vera que le langage reconnu par la grammaire G est le plus petit point fixe d'une fonction croissante. Pour cela il nous faudra introduire un peu d'analyse, de topologie sur les langages...

exercices

1. Montrer que: $(a \oplus a \odot a) \odot a$ est reconnu par la deuxième grammaire ci dessus.
2. Définir une grammaire acceptant la phrase suivante: le petit tabouret mange la purée.

3.2 Classification de Chomski

On a la classification suivante due a Chomski:

1. Langages de type 3: Ce sont les langages réguliers reconnus par des grammaires linéaires.
2. Langages de type 2: Ce sont les langages algébriques: les règles de productions doivent être de la forme $u \to v$ avec u $\in N$
3. Langages de type 1: Reconnus par des grammaires croissantes:$|u| \leq |v|$
4. Langages de type 0: Langages reconnus par n'importe quélles grammaires.

4 Les langages algébriques

4.1 Définition, exemples

D'après la classification de Chomski, Les langages algébriques sont ceux pouvant être engendrés par une grammaire algébrique soit: du deuxième type:

Définition d'une grammaire algébrique

Les règles de production doivent être de la forme $u \to v$ avec u $\in N$

Exemples, contres-exemples.

Les deux grammaires précedentes:

1. $N = \{S\}$, $T = \{a, b\}$, $P : S \mapsto aSb + \varepsilon$.

2. $N = \{E, T, F\}$, $T = \{a, \oplus, \odot, (,)\}$, $E : axiome$ et les productions:
   $E \mapsto E \oplus T + T$
   $T \mapsto T \odot F + F$
   $F \mapsto (F) + a$

sont des grammaires algébriques.

La grammaire ci-dessous avec:

$N = \{X, D, R\}$, $T = \{a, b, c\}$, et les productions:

$X \mapsto abc$
$X \mapsto aDbc$
$Db \mapsto bD$
$Dc \mapsto Rbcc$
$bR \mapsto Rb$
$aR \mapsto aaD$
$aR \mapsto aa$

n'est pas algébrique.

Exercice

Trouver le langage engendré par la grammaire ci-dessus.

5 outil mathématique : Le théorème de Knaster-Tarski

Il faut utiliser un théorème de convergence pour l'étude des ensembles ordonnés croissants. Ce théorème permet de montrer par exemple que l'algorihme de minimisation des automates finis converge par atteinte d'un plus petit point fixe. Ce théorème est aussi important pour trouver le langage reconnu par une grammaire: plus petite solution d'un système d'équations mais aussi pour obtenir des formes simplifiées, nettoyées de certaines grammaires . Un théorème permet d'associer à toute grammaire algébrique une grammaire engendrant le même langage mais s'adaptant mieux à l'analyse syntaxique: les formes normales de Greibach ressemblent aux grammaires lineaires droites, admettent une codification facile sous forme d'automates à piles .

5.1 Enoncé du théorème

Nous allons donner la définition suivante:

5.1.1 Ensemble ordonné complet

Un ensemble ordonné E est dit complet quand toute suite croissante admet une borne superieure c'est à dire:

$a_1 \leq a_2 \leq ...a_n...$

alors: $\exists a = \bigvee_i a_i$ Un tel ensemble est un $C.P.O$

Théoréme de Knaster-Tarski

Soit f continue de $E(C.P.O) \rightarrow E$. Si E contient un plus petit élément "bottom" noté: $\perp$, Alors f admet un plus petit point fixe.

Application continue

On dit que $f : E(C.P.O) \rightarrow F(C.P.O)$ est continue si: f croissante et

$f(\bigvee_i X_i) = \bigvee_i f(X_i)$

pour $X_1 \leq X_2 \leq ...X_n...$: suite croissante du $C.P.O$ E

Preuve du théorême

on a:

$\perp \leq f(\perp) \leq ...f^p(\perp)...$ car f croissante.

d'autre part: E complet donc il existe un élément y verifiant:

$y = \bigvee_i f^i(\perp)$

comme l'application f est continue il vient:

$f(y) = f(\bigvee_i f^i(\perp)) = \bigvee_i f^{i+1}(\perp) = y$ •.

exercice

1. Exprimer à l'aide du théorème ci dessus la minimisation des automates finis.
2. Proposer un algorithme permettant l'élimination des terminaux inaccessibles d'une grammaire algébrique.

6 Application du théorême de Knaster-Tarski

6.1 Langage engendré par une grammaire

on a le théorème suivant:

Théorème

Les langages algébriques engendré par une grammaire G sont les plus petites solutions de la grammaire vue comme système d'équations

Preuve

On définit la fonction

$\Phi : \mathcal{P}(T^*)^n \rightarrow \mathcal{P}(T^*)^n$

$(\mathrm{L}_1, ..., L_n) \rightarrow (L'_1, ..., L'_n)$

et la grammaire dont les règle de productions sont:

$X_1 = \alpha_{11} + \alpha_{12} + ... + \alpha_{1p_1}$

$X_2 = \alpha_{21} + \alpha_{22} + ... + \alpha_{2p_2}$

...

$X_n = \alpha_{n1} + \alpha_{n2} + ... + \alpha_{np_n}$

Φ est alors donnée par :

$\Phi(X_1, ..., X_n) = (\alpha_{11} + \alpha_{12} + ... + \alpha_{1p_1}, ..., \alpha_{n1} + \alpha_{n2} + ... + \alpha_{np_n})$

On montre facilement que f est croissante.

Que $\mathcal{P}(T^*)$, $\mathcal{P}(T^*)^n$ sont ordonnés complet.

Lemme

Φ continue.

En effet la concatenation et la somme (reunion) de deux langages est continue.

et cela termine la preuve du Théorème.•

6.2 Application à la résolution de systèmes

Pour trouver la solution d'un système d'équations définissant les règles de productions d'une grammaire on utilise en pratique l'algorithme fourni par le théorème suivant:

Lemme

Soit f la fonction de deux variables:

$f : E \times F \to E$

$(x, y) \mapsto f_y(x) = f(x, y)$

le plus petit point fixe en la variable x ($x = f(x, y)$: u(y) est continue.

Proposition

Méthode de résolution du système :

$x = f(x, y)$

$y = g(x, y)$

$\Phi : E \times F \to E \times F$

$(x, y) \mapsto (f(x, y), g(x, y))$

Déterminer le plus petit point fixe de:

$(\perp, \perp) \mapsto \Phi(\perp, \perp) \mapsto \Phi^2(\perp, \perp)$... . On utilise le lemme précedent : soit $u(y)$ la plus petite solution en x de la première équation, on reporte dans la seconde et soit y_1 la plus petite solution de $y = g(u(y), y)$,Alors $(u(y), y_1)$ est la plus petite solution du système.

Exemples

1. Considerons la première grammaire donnée:

   $N = \{S\}$,$T = \{a, b\}$, $P : S \mapsto aSb + \varepsilon$.

   ici $\perp = \emptyset$ et,

   $\Phi : \mathcal{P}(T^*) \to \mathcal{P}(T^*)$, $S \mapsto aSb + \varepsilon$ donc:

   $\Phi(\emptyset) = \{\varepsilon\}$

$\Phi^2(\emptyset) = \{\varepsilon, ab\}$

$\Phi^3(\emptyset) = \{\varepsilon, ab, a^2b^2\}$ ...

$\Phi^n(\emptyset) = \{\varepsilon, ab, ...a^{n-1}b^{n-1}\}$

La borne superieure existe dans l'ensemble des partie d'un ensemble et donnée par la reunion, ici:

$\bigvee_n \Phi^n(\emptyset) = \bigcup_n \Phi^n(\emptyset)$ soit:

$\bigvee_n \Phi^n(\emptyset) = \{a^n b^n / n \in N\}$

2. considerons la grammaire linéaire droite suivante sur $T = \{a, b, c, e\}$:

   $X_1 = (a + bc)X_1 + ae$

   $X_2 = bX_1 + (ce + a)X_2 + ab + c$

   En fait cette grammaire est régulière et on a un moyen pour trouver la solution autre qu'en passant par les équations de départ d'un automate fini.

   On utilise une variante "point fixe" du lemme d'Arden:

   la plus petite solution de $f(X) = LX + M$ est : L^*M.

   Ensuite on applique l'algorithme de résolution des systèmes vu plus haut: on cherche le plus petit point fixe en la variable X_1 et on reporte dans l'autre équation, on trouve:

   $X_1 = (a + bc)^*ae$

   $X_2 = (a + ce)^*(b(a + bc)^*ae + ab + c)$

6.3 Application au nettoyage de grammaires, formes normales de Chomski

Une application plus pratique encore du Théorème de Tarski consiste en le nettoyages de certaines grammaires par élimination de productions vides,de symboles inutiles de terminaux inaccéssibles,de cycles et productions simples afin de préparer des formes adécquates pour l'analyse syntaxiques: formes normales de Chomski et mieux formes normales de Greibach (objet du prochain paragraphe et qui contient une traduction immédiate en terme d'automates (à piles)) .

6.3.1 Exemple 1: Suppression des symboles inutiles

Il faut enlever les symboles X tel qu'il n'existe pas de derivations:

$X \rightarrow .... \rightarrow w \in T^*$

On considère alors la suite d'ensembles croissants suivante:

$N_1 = \{X \in N / \exists X \rightarrow w \text{ avec } w \in T^*\}$

$N_i = N_{i-1} \bigcup \{X \in N / \exists X \rightarrow \alpha \text{ avec des non terminaux de } N_{i-1} \text{ dans } \alpha\}$

On fait apparaitre la fonction croissante (et continue: l'ensemble E des non terminaux est fini):

$f(P) = P \bigcup \{X \in N / \exists X \rightarrow \alpha \text{ avec des non terminaux de } P \text{ dans } \alpha\}$

on cherche N_{i-1} point fixe de f: $(f(N_{i-1}) = N_i)$

Exemple

Considèrons la grammaire ci dessous:

$S \rightarrow ab + b + aTSb + U$

$V \rightarrow b$

$T \rightarrow bWb + \varepsilon$

$W \rightarrow S$

Si on calcul la suite d'ensembles croissants N_i, on aboutit rapidement à un point fixe en effet:

$N_1 = \{S, V, T\}$

$N_2 = \{S, V, T, W\}$

$N_3 = \{S, V, T, W\}$

$N_2 = N_3$: point fixe, Il faut donc éliminer le non terminal U: symbole inutile.

• Remarque : Il faut élaborer une autre suite d'ensembles croissants pour éliminer $V \rightarrow b$: inutile aussi

6.3.2 Exemple 2: Suppression des productions vides

Il faut modifier les non terminaux X tel que:

$X \rightarrow .... \rightarrow \varepsilon$ de la manière suivante:

On considère alors la suite d'ensembles croissants suivante:

$V_1 = \{X \in N / \exists X \rightarrow \varepsilon \in G\}$

$V_i = V_{i-1} \bigcup \{X \in N / \exists X \rightarrow \alpha$ avec des non terminaux de V_{i-1} dans $\alpha\}$

on considère la même fonction f que plus haut.

Ensuite on remplace dans la grammaire chaque non terminal X_i derivant sur ε par $X_i + \varepsilon$ dans la grammaire G

Exercice

Appliquer l'algorithme précedent à la grammaire examinée et nettoyée ci dessus.

6.3.3 Forme normale de Chomski

Une grammaire nettoyée peut être mise sous forme normale de Chomski:

6.3.4 Définition

Une grammaire est sous forme normale de Chomski (C.N.F) quand elle n'a que des productions de la forme:

$X \rightarrow AB$

$X \rightarrow a (\in T^*)$

$X \rightarrow \varepsilon$: pour l'axiome uniquement !

Proposition

Pour toute grammaire il existe une C.N.F equivalente.

Preuve

1. Il faut remplacer dans la grammaire nettoyée tout terminal a par une règle $X \rightarrow a$
2. Il faut remplacer la règle:

$X \to X_1 X_2 ... X_n$ par:

$X \to X_1 < X_2 ... X_n >$

$< X_2 ... X_n > \to X_2 < X_3 ... X_n >$ etc... •

Exercice

Trouver la C.N.F de la grammaire précedente à la nettoyée ci-dessous:

$S \to ab + b + aT'Sb + aSb$

$T' \to bSb$

•Remarque : c'est la grammaire nettoyée plus haut débarassée aussi des cycles et productions simples.

6.3.5 Utilitée de la forme normale de Chomski

Mise sous cette forme on peut décider par une "analyse montante" si un mot est engendré ou non par une grammaire: on doit retrouver l'axiome.

7 Forme normale de Greibach

On peut résumer en disant que l'on peut toujours nettoyer une grammaire algébrique et la mettre sous C.N.F. On va voir un autre algorithme transformant toute grammaire algébrique en forme de Greibach laquelle admet une traduction immédiate sous forme d'automate à pile.

Définition

Une grammaire est une forme normale de Greibach si ses productions sont de la forme:

$X \to a\alpha$ avec $a \in T$ et $\alpha \in N^*$ ou

$X \to \varepsilon$

en outre on exige que X n'appartient à aucun membre droit.

7.1 Mise sous forme de Greibach: algorithme

On donne la grammaire suivante avec les productions:

$X_1 = X_1\alpha_{11} + X_2\alpha_{12} + ... + X_n\alpha_{1n} + \beta_1$

$X_2 = X_1\alpha_{21} + X_2\alpha_{22} + ... + X_n\alpha_{2n} + \beta_2$

...

$X_n = X_1\alpha_{n1} + X_2\alpha_{n2} + ... + X_n\alpha_{nn} + \beta_n$

On peut ecrire ce système matriciellement:

$X = XA + B$ c'est le point fixe de l'application:

$f(X) = XA + B$.

En fait on double l' équation:

Le point fixe de f est BA^*. et $A^* = I + A^+$ avec

A^+ solution de $Q = AQ + A$ et $X = BA^* = B + BA^+$

Il faut donc résoudre le système d'équations matricielles:

$X = B + BQ$

$Q = AQ + A$

exercice

Mettre sous forme de Greibach la grammaire ci-dessous:

$X_1 \to a + X_2X_1$

$X_2 \to b + X_1X_2$

7.2 Forme normale de Greibach et automates a piles

Etant donnée une Grammaire de Greibach, on a la codification suivante sous forme d'automate à pile:

A toute règle $X \to aX_1X_2...X_n$

correspond: $X.a = X_n....X_1$

A toute règle $X \to a$

correspond: X.a = ε

References

[1] A.Aho, J. Ullman *The theory of parsing,translation and compiling*,Prentice-Hall, 1972.

[2] Clifford Preston, *Algebraic theory of semi-group*, Univ. Lecture series 6, A.N.S., 1961.

[3] E Eilenberg, *Automata, languages machines*, Academic press,1974.

[4] A. Salomaa, *Formal languages*, Academic press, 1972.