Langages algébriques & grammaires

Rationnels et grammaires linéaires

Questions :

- Peut on trouver des grammaires qui engendrent des langages rationnels ?
- Est-ce que tout langage rationnel peut être engendré par une grammaire ?

Rationnels et grammaires linéaires

- Une grammaire algébrique est dite linéaire (droite) si toutes ses règles de dérivation sont de la forme

  $X \to aY$

  $X \to a$

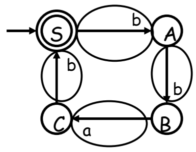
  $X \to \varepsilon$

  Avec $X,Y \in N$ et $a \in T$

Théorème : Si L est rationnel, L est engendré par une grammaire linéaire.

Exemple

$L=(bbab)^*$ rationnel $\Rightarrow$ il existe AFD qui le reconnaît



$T=\Sigma=\{a,b\}$

$N=Q=\{S, A, B, C\}$

S (axiome définie par l'état initial)

$S \to bA$

$A \to bB$

$B \to aC$

$C \to bS$

$S \to \varepsilon$

Car état de reconnaissance

Preuve

- L rationnel $\Rightarrow$ il existe AFD $A=(Q,\Sigma,\delta,i,F)$ tel que $L(A)=L$
- On construit $G=(N,T,S,R)$ à partir de M : $L(G)=L(A)=L$

  $N=Q$

  $T=\Sigma$

  $S=i$
- A chaque transition $\delta(p,a)=q$, on ajoute une production $P \to aQ$
- A chaque état $f \in F$, on ajoute une production $f \to \varepsilon$
- Reste à montrer que ça marche bien

Et ça marche !!!!

$(p,w) \Rightarrow^* (q,v)$ avec $w=uv$

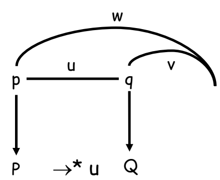
SSI

$P \to^* uQ$

- Par récurrence sur la longueur des dérivations
  - Vrai pour 1-dérivations:

  $(p,a) \Rightarrow (q,\varepsilon)$ SSI $P \to aQ$

  Par définition de G



Et ça marche !!!!

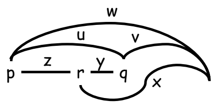
- (HR) Vrai pour des dérivations de longueur au plus n
- Soit une n+1-dérivation:
- $(p,w) \Rightarrow^{n+1}(q,v)$ avec $w=uv$

SSI $P \rightarrow^{n+1} uQ$

  - $(p,w) \Rightarrow^{n}(r,x) \Rightarrow^{1}(q,v)$ avec $w=zx$ et $x=yv$ SSI
  - $P \rightarrow^{n} zR$ et $R \rightarrow^{1} yQ$
  - $P \rightarrow^{n+1} zyQ$ et $zy=u$
- En particulier (pour terminer),

$(i,w) \Rightarrow^{*}(f,\varepsilon)$ SSI $i \rightarrow^{*} w$

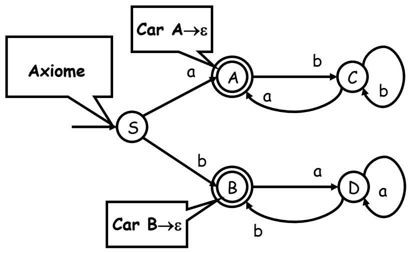
- *Ce qui implique que L(G)=L(A)*



Et réciproquement, étant donnée une grammaire linéaire, on peut construire un AFD?

Réciproque, exemple

- $S \rightarrow aA|bB;\ A \rightarrow bC|\varepsilon;\ B \rightarrow aD|\varepsilon;\ C \rightarrow bC|aA;\ D \rightarrow aD|bB$



- ici on a retrouvé un AFD correspondant à la grammaire linéaire.
- Est-ce vrai pour toute grammaire linéaire ?

Réciproque

Théorème : Si un langage est engendré par une grammaire linéaire alors il est rationnel

- L est engendré par $G=(N,T,S,R)$; on construit un AFND $A=(\Sigma = T, Q = N \cup \{f\}, \delta, i = S, T=\{f\})$ tel que $L(A)=L(G)$. La fonction non déterministe δ est
  - à chaque règle de la forme $X \rightarrow wY$ on introduit une transition de la forme $\delta(X,w)=Y$
  - à chaque règle de la forme $X \rightarrow w$ une transition $\delta(X,w)=f$
- il faudrait montrer par récurrence sur la longueur des dérivations que $L(G)=L(A)$ (raisonnement analogue au précédent).

Remarques & conclusion

- *Observons que dans la démonstration on n'a pas de règle qui donne le mot vide.*
- *On verra qu'il est toujours possible de trouver une grammaire sans ε-production qui engendre le même langage (sauf si ε appartient au langage).*

Théorème :

Un langage est rationnel ssi il est engendré par une grammaire linéaire droite

Les grammaires linéaires gauches

- linéaire droite: productions de la forme

$X \rightarrow aY|a|\varepsilon$ avec $X,Y \in N$ et $a \in T$

- linéaire gauche: productions de la forme

$X \rightarrow Ya|a|\varepsilon$ avec $X,Y \in N$ et $a \in T$

Intérêt des grammaires linéaires

- Soit la grammaire (qui n'est pas linéaire)
  - Exp →var|cst
  - Exp →Exp * Exp| Exp + Exp
- On peut lui associer une expression rationnelle

(var|cst) [(+|*) (var|cst)]*

Et trouver une grammaire linéaire équivalente

décimaux JAVA BNF

- DecimalNumeral: 0|NonZeroDigit [Digits] N
- Digits:Digit|Digits Digit D
- Digit: 0| NonZeroDigit A
- NonZeroDigit: one of 1 2 3 4 5 6 7 8 9 C

- $C=\{1,2,3,4,5,6,7,8,9\}$ $A=C\cup\{0\}$
- $D=A^+$
- $N=0+CD+C=0+CA^*$

DecimalNumeral=0+(1+2+3+4+5+6+7+8+9)(0+1+2+3+4+5+6+7+8+9)*

13

Rationnels et compilation

- Si une partie des langages informatiques est rationnelle (et cela est fort utile pour la compilation), ce n'est hélas pas toujours le cas
- L'exemple le plus simple d'une partie de langage informatique non rationnelle est celui des mots de Dyck :

$$G=\langle N=\{S\},T=\{(,)\},R=\{S\to(S)|SS|\varepsilon\},S\rangle;\ L(G)=D$$

- D n'est pas rationnel :
  - Il suffit de considérer $D\cap(^*)^* = (^n)^n$
  - $(^*)^*$ est un langage rationnel
  - $D\cap(^*)^*$ devrait être rationnel (propriétés de clôture)
  - Or $(^n)^n$ n'est pas rationnel (c'est a^nb^n)
  - Donc D n'est pas rationnel

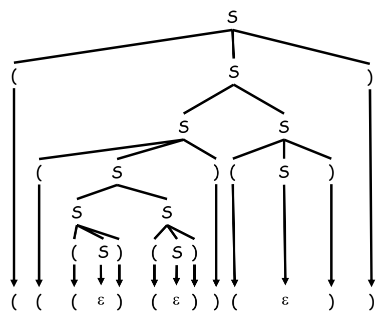
14

Exemple d'EBP*

*Expression Bien Parenthésé

$R=\{S\to(S)|SS|\varepsilon\}$

(((())())())



15

Langages & grammaires algébriques

- C'est ce qui motive l'étude des langages algébriques (ou non contextuels)
- Ils sont indispensables à la réalisation des langages informatiques
- Toujours pour des motivations informatiques, il est nécessaire d 'avoir des grammaires algébriques aussi simples que possible

- Objectif de la simplification des grammaires

16

Nouvelle notion

- On dit que deux grammaires sont équivalentes si elles engendrent le même langage

$$G\approx G' \Leftrightarrow L(G)=L(G')$$

Exemple : $\{a^nb^n : n\geq 0\}$

- engendré

soit par

$$S\to aSb|\varepsilon$$

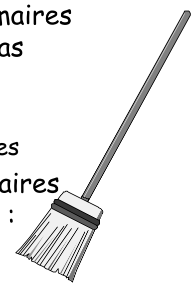
soit par

$$S\to aSb|ab\ |\varepsilon$$

17

Motivation de la simplification

- Il s'agit de « nettoyer » les grammaires pour en retirer tout ce qui n'est pas strictement indispensable à la génération des mots du langage :
  - Retirer les variables et règles inutiles
- on peut ensuite mettre les grammaires sous des formes standard simples :



Les formes normales

18

Idées

- Comment modifier les grammaires sans pour autant en restreindre l'« expressivité »?
- Un langage algébrique non vide L peut être engendré par une grammaire G vérifiant :
  - Chaque variable doit mener à la génération d'un mot de L **(variable productif)**
  - Chaque variable doit servir à quelque chose, donc on doit pouvoir le retrouver lors d'une dérivation

    (variable accessible)

19

Supprimer les improductifs

- $X \in N$ est productif s'il existe $w \in T^*$ tq $X \rightarrow^* w$
- Si X n'est pas productif, cela signifie qu'on peut supprimer X sans retirer de mots au langage engendré par la grammaire.
- On construit inductivement P, l'ensemble des variables productives :
  - Base : $P_0 = \emptyset$
  - Règle : $P_{i+1} = \{X \in N : X \rightarrow \alpha,\ \alpha \in (T \cup P_i)^*\}$

  On s'arrête lorsque $P_{i+1} = P_i = P$

Lemme : Étant donnée $G=(N,T,S,R)$ t.q. $L(G) \neq \emptyset$, on peut trouver une grammaire équivalente $G'=(N',T,S,R')$ sans symboles improductifs.

20

Exemple

- Soit la grammaire
  - $S \rightarrow AB|a,\ A \rightarrow a,\ B \rightarrow BA$

$P_0 = \emptyset$

$P1=\{X \in N : X \rightarrow \alpha,\ \alpha \in T^*\}=\{S,A\}$

$P2=\{X \in N : X \rightarrow \alpha,\ \alpha \in (T \cup \{S,A\})^*\}=\{S,A\}$

P1 =P2 . On en déduit que B est improductif

On supprime les règles contenant B : ie $S \rightarrow AB$ et $B \rightarrow BA$ pour obtenir la grammaire équivalente $S \rightarrow a,\ A \rightarrow a$

21

Démonstration (1)

- Par récurrence sur n, longueur de la dérivation
  - Base : n=1. Pour toute production $A \rightarrow w,\ w \in T^*$, on ajoute A à P_1.
  - Induction : $A \rightarrow X_1X_2\ldots X_m \rightarrow^* w$ en k étapes. Alors $w=w_1w_2\ldots w_m$ où $X_i \rightarrow^* w_i$ pour $0<i<m+1$ par des dérivations de moins de k étapes.
    Par HR, chacun des $X_i \in P_{k-1}$ et, par définition de P_k, A est ajouté dans P_k.

22

Démonstration (2)

- Comme $P_i \subseteq P_{i+1}$, nécessairement il arrive que $P_i = P_{i+1}$. En effet, P_i ne peut en aucun cas contenir plus de non terminaux que N lui même, cas où tous les non terminaux sont productifs.
- $N'=P$ et les symboles des règles de R' sont tous dans $(N' \cup T)^*$.
- $G' = (N'=P,T,S,R')$ satisfait la propriété :

  si $A \in N'$, alors $A \rightarrow^* w$

23

Équivalence des grammaires

- Comme toute dérivation de G' est une dérivation de G

  $L(G') \subseteq L(G)$

- Et si l'inclusion était stricte ?

  Si $w \in L(G) \setminus L(G')$ alors toute dérivation de w dans G utilise un non terminal de $V \setminus V'$ ou une production de $R \setminus R'$. En ce cas, il existe un non terminal de $V \setminus V'$ qui permet de dériver une chaîne terminale, contradiction

24

Algorithme

Début

Ancien_N $\leftarrow \varnothing$

Nouveau_N $\leftarrow \{A : A \rightarrow w$ pour $w \in T^*\}$

tantque Ancien_N $\neq$ Nouveau_N

Ancien_N $\leftarrow$ Nouveau_N

Nouveau_N $\leftarrow$ Ancien_N $\cup \{A : A \rightarrow \alpha, \alpha \in (N \cup \text{Ancien\_N})^*\}$

fintantque

N':=Nouveau_N

Fin

Retour à l'exemple

- Pour la grammaire

  $S \rightarrow AB|a,\ A \rightarrow a,\ B \rightarrow BA$

- En supprimant les symboles improductifs on a obtenu la grammaire équivalente

  $S \rightarrow a,\ A \rightarrow a$

- Mais à quoi sert la production $A \rightarrow a$?

Supprimer les inaccessibles

- $X \in N$ est accessible s'il existe α et $\beta \in (N \cup T)^*$ tq $S \rightarrow^* \alpha X \beta$
- Si X n'est pas accessible, cela signifie qu'on peut supprimer X sans retirer de mots au langage engendré par la grammaire.
- Pour trouver les variables accessibles, il suffit de parcourir les règles en partant de l'axiome (cailloutage)

Lemme : Étant donnée $G=(N,T,S,R)$, on peut trouver une grammaire équivalente $G'=(N',T,S,R')$ sans symbole inaccessible.

Exemple

$S \rightarrow aAb|a$

$A \rightarrow aAC|b$

$B \rightarrow d$

$C \rightarrow aSbS|aba$

Variables accessibles : $\{S,A,C\}$

Démonstration

L'ensemble $N' \cup T'$ de symboles accessibles de G est obtenu au moyen d'un algorithme itératif.

On met S dans N'.

Si A est dans N', et $A \rightarrow \alpha_1 \alpha_2 \ldots \alpha_n$ alors on ajoute à N' tous les non terminaux de $\alpha_1, \alpha_2, \ldots, \alpha_n$ et à T' tous les terminaux de $\alpha_1, \alpha_2, \ldots, \alpha_n$. R' est alors l'ensemble des productions de R qui ne contiennent que des symboles de $N' \cup T'$.

Nettoyage de grammaires

- En appliquant les deux lemmes précédents, on peut transformer une grammaire algébrique en une grammaire équivalente qui ne contient pas de symbole inutile (improductif ou inaccessible).
- Observons que si on retire tout d'abord les inaccessibles puis les improductifs, on ne retire pas forcément l'ensemble des symboles inutiles.
- **Ordre d'application** :
  1. Retirer les improductifs
  2. Retirer les inaccessibles

Exemple du mauvais ordre

- $S \to aAb|bAB|a$
- $A \to aAC$
- $B \to d$
- $C \to aSbS|aba$

On retire tout d'abord les inaccessibles
Tous les symboles non terminaux sont accessibles

Exemple du mauvais ordre

$S \to aAb|bAB|a$
$A \to aAC$
$B \to d$
$C \to aSbS|aba$

$S \to a$

inaccessibles ! $\left\{ \begin{array}{l} B \to d \\ C \to aSbS|aba \end{array} \right.$

équivalente

On retire tous les symboles improductifs :

- $P_0=\emptyset$
- $P_1=\{X\in N:N\to a,\ a\in T\}=\{S,B,C\}$
- $P_2=\{X\in N:N\to \alpha,\ \alpha\in (\{S,B,C\}\cup T)^*\}=\{S,B,C\}= P_1$
- A est donc improductif

Théorème

Théorème : Tout langage algébrique peut être engendré par une grammaire algébrique qui ne contient pas de symboles inutiles (improductifs ou inaccessibles)

- *Conséquence directe de l'application des deux lemmes précédents.*