Analyse de Données Structurées - Cours 4

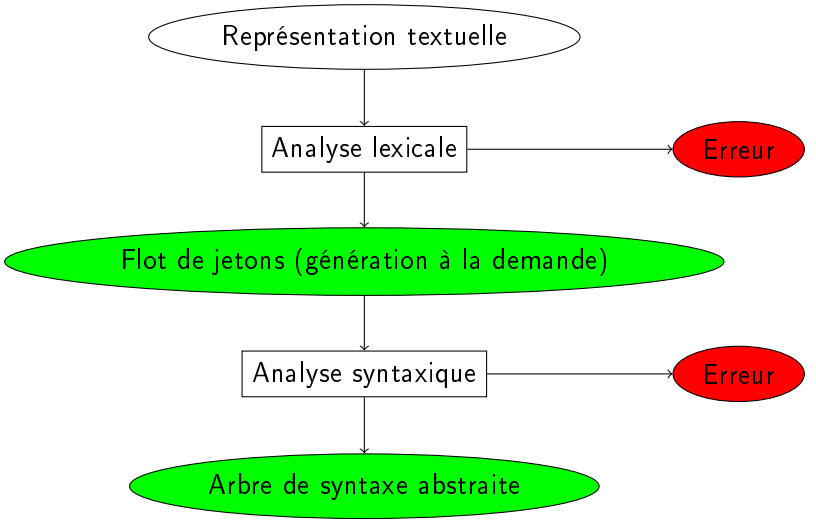
Ralf Treinen

Université Paris Diderot
UFR Informatique
Laboratoire Preuves, Programmes et Systèmes

treinen@pps.univ-paris-diderot.fr

11 février 2015

Le rôle de l'analyse syntaxique



Objectif de l'analyse syntaxique

- Deux objectifs :
  - Détecter des textes d'entrée qui ne sont pas correctement formés (et donner des indications utiles sur la nature de l'erreur).
  - Si l'entrée est correcte, construire un arbre de syntaxe abstraite.
- Focalisons d'abord sur le premier objectif : distinguer des textes corrects des textes incorrects.

Pourquoi deux étapes séparées ?

- On essaye de faire tant d'analyse que possible dans l'analyse lexicale.
- Raison : l'exécution d'un automate est très efficace (temps linéaire dans la longueur du texte d'entrée).
- Problème : l'expressivité des automates est limitée (voir les transparents suivants).

Reconnaître les expressions arithmétiques

- L'ensemble des expressions arithmétiques correctement parenthésées n'est pas régulier.
- Nous démontrerons une simplification :
  l'ensemble des mots formés seulement de parenthèses '[' et ']' qui sont correctement imbriquées, n'est pas régulier.
- Exemple d'un mot correctement imbriqué :

  [[][[]]]

- Exemple d'un mot qui n'est pas correctement imbriqué :

  []][

Reconnaître l'imbrication correcte des parenthèses

Définition du langage P

P est le plus petit langage tel que

- $\epsilon \in P$
- si $x, y \in P$ alors $xy \in P$
- si $x \in P$ alors $[x] \in P$

Théorème

P n'est pas régulier.

Un lemme

Lemme :

Le langage $L = \{[^n]^n \mid n \geq 1\}$ n'est pas régulier.

Intuition

- L consiste en tous les mots de la forme

  $$\underbrace{[\cdots[}_{n}\ \underbrace{]\cdots]}_{n}$$

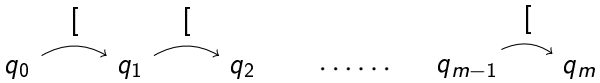
  pour un $n \geq 1$ quelconque.
- Un automate qui accepte L devrait compter les '[', et puis comparer ce nombre avec le nombre des ']'.
- Or, un automate fini ne peut pas compter une quantité non bornée.

Du lemme au théorème

- Avant de démontrer le lemme, réfléchissons pourquoi le lemme implique le théorème :
- Supposons pour l'absurde que P soit régulier.
- Le langage $L' = \{[^n]^m \mid n, m \geq 0\}$ est régulier.
- $L = P \cap L'$
- L'intersection de deux langages régulier est régulier : contradiction !

La preuve du lemme

- Supposons, pour l'absurde, que l'automate $A = (Q, q_o, F, \Delta)$ accepte L. Soit m le nombre d'éléments de Q.
- A accepte donc le mot $[^m]^m$.
- Regardons les actions de l'automate quand il lit m fois le symbole '[' :

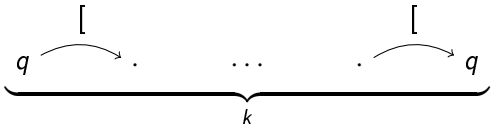


- Ça fait $m+1$ états dans cette séquence, mais il n'y a que m états différents.

Analyse de Données Structurées - Cours 4
└Analyse Syntaxique

La preuve du lemme

- Donc il y a un état, disant q qui parait deux fois dans cette séquence :

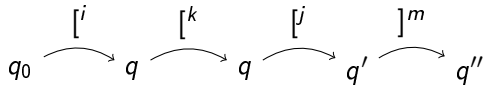


- Soit k la longueur de cette séquence. Nous avons donc $\Delta^*(q, [^k) = q$.

Analyse de Données Structurées - Cours 4
└Analyse Syntaxique

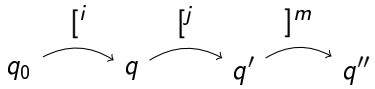
La preuve du lemme

- On a donc pour le mot $[^m]^m$:



avec $m = i + k + j$.

- Donc, on a aussi :



- Donc : $[^{m-k}]^m \in \mathcal{L}(A)$, ce qui est absurde !

Analyse de Données Structurées - Cours 4
└Analyse Syntaxique

Le résultat général

Lemme de l'étoile (angl. : *pumping lemma*)

Pour tout langage régulier L il existe un entier m tel que tout mot $w \in L$ de longueur $|w| \geq m$ a une décomposition $w = xyz$ tel que

- $0 < |y|$
- $|xy| \leq m$
- $xy^n z \in$ pour tout $n \geq 0$

Démonstration

Comme sur l'exemple des transparents précédents.

Définition des grammaires algébriques

Une *grammaire algébrique* (où : *grammaire hors contexte*) est un tuple $G = (V_T, V_N, S, R)$ où

- V_T est un ensemble fini de symboles dits *terminaux* ;
- V_N est un ensemble fini de symboles *non-terminaux*, où $V_N \cap V_T = \emptyset$;
- $S \in V_N$ est l'*axiome* ;
- R est un ensemble fini de *règles* de la forme $N \to u$, où $N \in V_N$, et $u \in (V_N \cup V_T)^*$.

Exemple d'une grammaire algébrique

$G_1 = (V_T, V_N, S, R)$ où

- $V_T = \{0, 1, +, *\}$
- $V_N = \{\mathrm{E}, \mathrm{I}\}$
- $S = \mathrm{E}$
- R consiste en les règles suivantes :

$$\begin{aligned} \mathrm{E} &\to \mathrm{E} + \mathrm{E} \\ \mathrm{E} &\to \mathrm{E} * \mathrm{E} \\ \mathrm{E} &\to \mathrm{I} \\ \mathrm{I} &\to 0 \\ \mathrm{I} &\to 1 \end{aligned}$$

Dérivation

Réécriture

Étant donnée une grammaire $G = (V_T, V_N, S, R)$, on note $V = V_T \cup V_N$.
Le mot $u \in V^*$ se *réécrit* en le mot $v \in V^*$ dans G, noté $u \to v$, si

1. $u = w_1 N w_2$, où $w_1 \in V^*$, $N \in V_N$ et $w_2 \in V^*$;
2. $N \to w$ est une règle de R ;
3. $v = w_1 w w_2$.

Dérivation

Le mot $v \in V^*$ *dérive* du mot $u \in V^*$, dans la grammaire G, noté $u \to^* v$, s'il existe une suite finie $w_0, w_1, \ldots, w_n$ de mots de V^* telle que $w_0 = u$, $w_i \to w_{i+1}$ pour tout $i \in [0, n-1]$, et $w_n = v$.

Exemple

- Pour la grammaire G donnée au dessus on a les réécritures suivantes :
  (Nous mettons en rouge le non-terminal qui est réécrit.)
  - $\mathrm{E} \to \mathrm{E} + \mathrm{E}$
  - $\mathrm{E} + \mathrm{E} \to \mathrm{E} + \mathrm{E} + \mathrm{E}$
  - $\mathrm{E} + \mathrm{E} + \mathrm{E} \to \mathrm{E} + \mathrm{E} + \mathrm{E} * \mathrm{E}$
  - $\mathrm{E} * \mathrm{E} + \mathrm{E} * \mathrm{E} \to \mathrm{E} * \mathrm{E} + \mathrm{I} * \mathrm{E}$

Exemple : Dérivation de 1+0*1 dans G_1

$$
\begin{aligned}
E &\rightarrow E + E \\
&\rightarrow E + E * E \\
&\rightarrow I + E * E \\
&\rightarrow I + E * I \\
&\rightarrow I + E * 1 \\
&\rightarrow 1 + E * 1 \\
&\rightarrow 1 + I * 1 \\
&\rightarrow 1 + 0 * 1
\end{aligned}
$$

Langage engendré

Définition

Soit $G = (V_T, V_N, S, R)$ une grammaire algébrique. Le *langage engendré* par G est

$$\mathcal{L}(G) = \{w \in V_T^* \mid S \rightarrow^* w\}$$

Un langage est *algébrique* s'il est engendré par une grammaire algébrique.

Exemple

Pour la grammaire G_1 on obtient que $\mathcal{L}(G_1)$ est l'ensemble des expressions arithmétiques formées à l'aide des opérateurs + et *, et des constantes 0 et 1, et sans parenthèses.

Relation avec les langages régulières

- Tout langage régulier est algébrique.
- Par contre, il y a des langages algébriques qui ne sont pas réguliers : $G_P = (\{[,]\}, \{E\}, E, R)$ où R consiste en

$$
\begin{aligned}
E &\rightarrow \epsilon \\
E &\rightarrow E\ E \\
E &\rightarrow [\ E\ ]
\end{aligned}
$$

On a que $\mathcal{L}(G_P) = P$, le langage des séquences de parenthèses correctement imbriquées.

Notation : alternatives

- Une grammaire peut avoir plusieurs règles avec le même côté gauche :

$$
\begin{aligned}
E &\rightarrow E + E \\
E &\rightarrow E * E
\end{aligned}
$$

- Nous permettons dans la suite dans ce cas d'écrire une seule règle, avec plusieurs *alternatives* sur le côté droite :

$$E \rightarrow E + E \mid E * E$$

Dérivation gauche

Soit une grammaire $G = (V_T, V_N, S, R)$; on note $V = V_T \cup V_N$.

Réécriture à gauche

Le mot $u \in V^*$ se *réécrit à gauche* en le mot $v \in V^*$ dans G, noté $u \rightarrow_g v$, si

1. $u = w_1 N w_2$, où $w_1 \in V_T^*$, $N \in V_N$ et $w_2 \in V^*$;
2. $N \rightarrow w$ est une règle de R ;
3. $v = w_1 w w_2$.

Dérivation gauche

Une *dérivation gauche* est une suite finie $w_0, w_1, \ldots, w_n$ de mots de V^* telle que $w_i \rightarrow_g w_{i+1}$ pour tout $i \in [0, n-1]$.

Exemple : Dérivation gauche de 1+0*1 dans G_1

$$
\begin{aligned}
E &\rightarrow E * E \\
&\rightarrow E + E * E \\
&\rightarrow I + E * E \\
&\rightarrow 1 + E * E \\
&\rightarrow 1 + I * E \\
&\rightarrow 1 + 0 * E \\
&\rightarrow 1 + 0 * I \\
&\rightarrow 1 + 0 * 1
\end{aligned}
$$

Arbre de dérivation

Définition

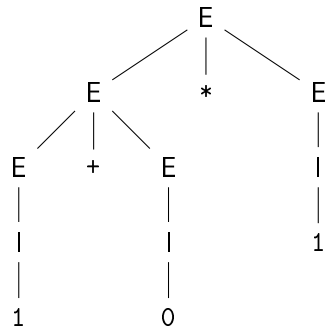
Soit $G = (V_T, V_N, S, R)$ une grammaire. Un *arbre de dérivation* de G est un arbre tel que

- les nœuds internes sont étiquetés par des symboles de V_N ;
- les feuilles sont étiquetées par des symboles de $V_T \cup V_n$;
- si les fils pris de gauche à droite d'un nœud interne étiqueté par le non-terminal N sont étiquetés par les symboles respectifs $\alpha_1, \ldots, \alpha_n$, alors $(N \rightarrow \alpha_1 \cdots \alpha_n) \in R$.

Définition

Un arbre de dérivation dont la racine est étiquetée par l'axiome de la grammaire et dont le mot des feuilles u appartient à V_T^* est appelé *arbre de dérivation de u*.

Exemple : Un arbre de dérivation de 1+0*1 dans G_1



D'une dérivation gauche à l'arbre de dérivation

- La racine de l'arbre est étiquetée par l'axiome de la grammaire (qui est aussi le premier élément de la dérivation gauche).
- On parcourt la dérivation de gauche à droite, en complétant l'arbre.
- À la réécriture du premier non-terminal dans un mot correspond l'extension de la feuille de l'arbre de dérivation en construction la plus gauche qui est étiquetée par un non-terminal.
- (Voir l'exemple donné au tableau)

De l'arbre de dérivation à la dérivation gauche

- Le premier élément de la dérivation gauche est l'axiome de la grammaire (qui est aussi l'étiquette de la racine de l'arbre).
- On fait un parcours préfixe de l'arbre, en complétant la dérivation gauche.
- Aux fils d'un nœud interne correspond la réécriture du premier non-terminal dans un élément de la dérivation.
- (Voir l'exemple donné au tableau)

Arbres de dérivation et Dérivation gauches

- Équivalence entre dérivations gauches et arbres de dérivation :
  - Pour chaque arbre de dérivation il y une unique dérivation gauche.
  - Pour chaque dérivation gauche il y a un unique arbre de dérivation.
- Dans un premier temps, on peut imaginer l'arbre de dérivation comme résultat de l'analyse syntaxique.
- Il nous faut donc que chaque mot du langage possède un arbre de dérivation unique.

Grammaires ambiguës

Définition

Une grammaire G est non-ambiguë quand tout $w \in \mathcal{L}(G)$ a un seul arbre de dérivation.

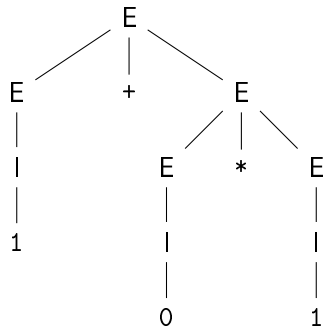
Définition équivalente

Une grammaire G est non-ambiguë quand tout $w \in \mathcal{L}(G)$ a une seule dérivation gauche.

Sur l'exemple G_1

La grammaire G_1 est ambiguë : le mot `1+0*1` a deux arbres de dérivation différents !

Un autre arbre de dérivation de 1+0*1 dans G_1



Un autre exemple

- La grammaire $G_2 = (\{\mathtt{i}, \mathtt{v}, +, *, (,)\}, \{\mathsf{E}\}, \mathsf{E}, R)$, où R est

$$\mathsf{E} \rightarrow \mathtt{i} \mid \mathtt{v} \mid (\mathsf{E} + \mathsf{E}) \mid (\mathsf{E} * \mathsf{E})$$

- Cette grammaire décrit les expressions arithmétiques complètement paranthésées.
- Ici il y a un seul terminal `i` pour les entiers, et un seul terminal `v` pour les noms des variables car en imagine qu'il s'agit des jetons issus d'une analyse lexicale.

G_2 est non-ambiguë

- Pour le démontrer nous avons besoin de deux petits lemmes.
- Utilité de ces lemmes : Imaginez qu'un mot w à deux dérivations gauches qui sont de la forme

$$\mathsf{E} \rightarrow (\mathsf{E} + \mathsf{E}) \rightarrow^* w$$
$$\mathsf{E} \rightarrow (\mathsf{E} * \mathsf{E}) \rightarrow^* w$$

- Soit w_1 le mot dérivé du E gauche de la première ligne, w_2 le mot dérivé du E gauche de la deuxième ligne.
- Forcement, $w_1 \neq w_2$. Donc, $w_1 <_{\text{pref}} w_2$ (ou l'inverse).
- Les lemmes nous permettent de démontrer que c'est impossible d'avoir $w_1, w_2 \in \mathcal{L}(G_2)$ avec $w_1 <_{\text{pref}} w_2$.

Séparer $\mathcal{L}(G_2)$ et ses propres préfixes

Notation

- $|w|_($ est le nombre de symboles (dans le mot w.
- $|w|_)$ est le nombre de symboles) dans le mot w.

Lemme 1

Pour tous $w \in \mathcal{L}(G_2)$: $|w|_(= |w|_)$.

Démonstration

Par induction sur la longueur de la dérivation la plus courte de w (au tableau).

Séparer $\mathcal{L}(G_2)$ et ses propres préfixes

Lemme 2
Pour tous $w \in \mathcal{L}(G_2)$: pour tous $v <_{\text{pref}} w$ avec $v \neq \epsilon$:
$|v|_(> |v|_)$.

Démonstration
Par induction sur la longueur de la dérivation la plus courte de w (au tableau).

G_2 est non-ambiguë

- Supposons pour l'absurde que G_2 est ambiguë.
- Soit w un mot de longueur minimale qui a deux dérivations gauches.
- Il n'est pas possible qu'une des deux commence sur $\mathrm{E} \to \mathtt{i}$ ou $\mathrm{E} \to \mathrm{v}$.
- Donc, une commence sur $\mathrm{E} \to (\mathrm{E} \circ \mathrm{E})$, et l'autre sur $\mathrm{E} \to (\mathrm{E} \diamond \mathrm{E})$, avec $\circ, \diamond \in \{+, *\}$.
- Donc :

$$\begin{aligned} w &= (v_1 \circ v_2) \\ w &= (u_1 \diamond u_2) \end{aligned}$$

avec $\mathrm{E} \to^* v_1, v_2, u_1, u_2$, de longueurs plus petites que w. Chacun de ces mots a donc une seule dérivation gauche !

G_2 est non-ambiguë

- Soit $v_1 = u_1$, alors forcement $\circ = \diamond$, $v_2 = u_2$, et les deux dérivations gauches sont les mêmes. Contradiction !
- Soit $v_1 <_{\text{pref}} u_1$. On a donc $v_1 \in \mathcal{L}(G_2)$, et $v_1 <_{\text{pref}} u_1 \in \mathcal{L}(G_2)$: Contradiction aux Lemmes 1 et 2 !
- Soit $u_1 <_{\text{pref}} v_1$. On a donc $u_1 \in \mathcal{L}(G_2)$, et $u_1 <_{\text{pref}} v_1 \in \mathcal{L}(G_2)$: Contradiction aux Lemmes 1 et 2 !

Backus-Naur Form

- Une notation très utilisée pour la documentation des langages de programmation est la notation *BNF* (Backus-Naur Form) qui est équivalente aux grammaires algébriques.
- Les non-terminaux sont écrit entre chevrons : $\langle T \rangle$.
- La flèche est remplacée par `::=`
- Les alternatives pour le même non-terminal sont regroupées comme indiquées au-dessus.
- Exemple :
  `<cond> ::= if "(" <condition> ")" "{" <code> "}"`

Extended Backus-Naur Form

- La forme étendue permet aussi des constructions des expressions régulières dans la grammaire : opérateurs $*$ et $+$, et parenthèses.
- Une partie entre crochets [ ] est optionnelle.
- Exemple : `<explist> ::= "(" <exp> ("," <exp>)* ")"`
- C'est encore équivalent aux grammaires algébriques.

Diagramme de syntaxe (non-récursif)

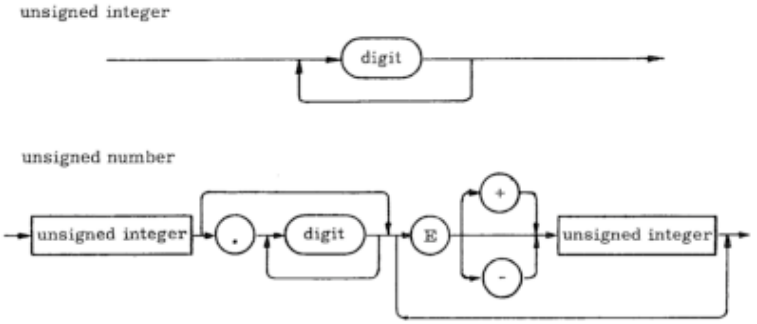


Diagramme de syntaxe (récursif)