Traduction et Sémantique
Analyse syntaxique

Luigi Santocanale
LIF, Université de Provence
Marseille, FRANCE

12 février 2010

Plan

Plan

Une grammaire non contextuelle . . .

. . . est un tuplet

$$\mathscr{G} = \langle V, \Sigma, S, P \rangle$$

où :

- V est un alphabet fini,
- $\Sigma \subseteq V$ est l'alphabet des symboles *terminaux*,
- $S \in V \setminus \Sigma$ est le *symbole de départ*,
- $P \subseteq (V \setminus \Sigma) \times V^*$ est un ensemble fini de *productions*.

On appelle un symbole $X \in V \setminus \Sigma$ un *non-terminal*.

Un exemple

- $V = \{S, T, a, b\}$,
- $\Sigma = \{a, b\}$,
- et $P = \{(S, T), (T, TT), (T, aTb), (T, \varepsilon)\}$,
  ce qu'on écrit :

$$S \to T$$
$$T \to TT$$
$$T \to aTb$$
$$T \to \varepsilon$$

ou, alternativement :

$$S \to T$$
$$T \to TT \mid aTb \mid \varepsilon .$$

« Dériver »

Soient $w, v \in V^*$.

- On écrit

$$w \Rightarrow v$$

  si
  - $w = w_0 X w_1$,
  - $v = w_0 u w_1$, et
  - la grammaire contient la production

$$X \to u .$$

- On écrit

$$w \Rightarrow^* v \qquad \text{(à lire : } w \text{ se dérive à } v\text{)}$$

  s'il existe $n \geq 0$ et $w_0, w_1, \ldots, w_n$ tels que :
  - $w = w_0$, $w_n = v$, et
  - $w_0 \Rightarrow w_1 \Rightarrow \ldots w_{n-1} \Rightarrow w_n$.

Langage d'une grammaire

Défini par :

$$L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \} .$$

Attention :
si $w \in L(\mathscr{G})$, alors chaque lettre de w est un symbole terminal.

Une *dérivation* de $w \in \Sigma^*$ de S
est une suite $w_0, \ldots, w_n$ telle que :

- $S = w_0$, $w_n = v$, et
- $w_0 \Rightarrow w_1 \Rightarrow \ldots w_{n-1} \Rightarrow w_n$.

Un exemple

Le mot $abab \in L(\mathscr{G})$, où $\mathscr{G}$ est la grammaire

$$S \to T$$
$$T \to TT$$
$$T \to aTb$$
$$T \to \varepsilon$$

car :

$$S \Rightarrow T \Rightarrow TT \Rightarrow aTbT \Rightarrow aTbaTb \Rightarrow aTbab \Rightarrow abab .$$

En fait :

$$\begin{aligned} L(G) &= \{ w \in \Sigma^* \mid w \text{ est bien parenthésé} \} \\ &= \{ w \in \Sigma^* \mid |w|_a = |w|_b \\ &\qquad \text{et } |u|_b \leq |u|_a \text{ si } u \text{ est un préfixe de } w \} . \end{aligned}$$

Grammaires sans ε-règles

Une ε-règle est une production de la forme

$$T \to \varepsilon .$$

Lemme. Toute grammaire $\mathscr{G}$ peut se transformer en une grammaire $\mathscr{G}'$ telle que toute ε-règle a la forme :

$$S \to \varepsilon$$

(où S est le symbole de départ).

Notre exemple :

$$S \to \varepsilon \mid T$$
$$T \to TT \mid aTb \mid ab .$$

Nous considérerons des grammaires ayant cette forme.

Dérivation gauche et droite

- On appelle une *dérivation gauche* si on récrit toujours le non-terminal plus à gauche.
- On appelle une *dérivation droite* si on récrit toujours le non-terminal plus à droite.
- Un mot $w \in L(\mathscr{G})$ peut avoir *plusieurs dérivations* qui témoignent de son appartenance à $L(\mathscr{G})$.

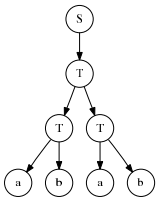
Arbre de dérivation

On utilse les productions de $\mathscr{G}$pour construire un arbre.

Soit $\mathscr{G}$:

$$S \to \varepsilon \mid T$$
$$T \to TT \mid aTb \mid ab .$$

$S \Rightarrow T$
$\Rightarrow TT$
$\Rightarrow Tab$
$\Rightarrow abab$



Le mot qui apparaît sur les feuilles est alors un mot de $L(\mathscr{G})$.

Dérivations et leurs arbres

- Il existe une bijection entre dérivations gauches d'un mot w et arbres de dérivation de w.
- Il existe une bijection entre dérivations droites d'un mot et ses arbres de dérivation.
- Un mot peut avoir plusieurs arbres de dérivation, « essentiellement différentes ».
- Un mot peut avoir plusieurs arbres de dérivation.

Ambiguïté

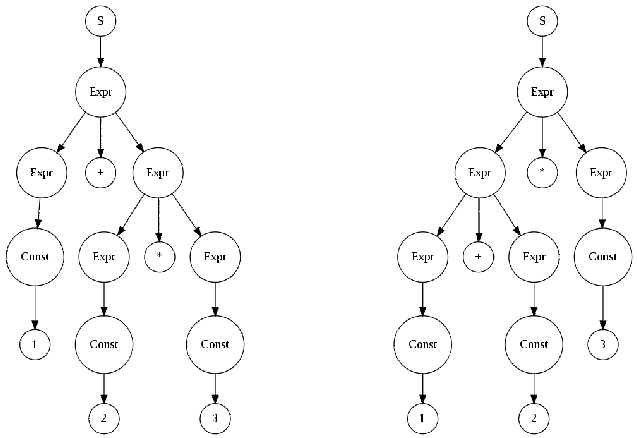
- Un *mot* $w \in L(\mathscr{G})$ est *ambigu*
  s'il possède plus qu'un arbre de dérivation.
- Une *grammaire* $\mathscr{G}$ est *ambiguë*
  s'il existe un mot $w \in L(\mathscr{G})$ qui est ambigu.
- Un *langage* L est *ambigu*
  si toute grammaire $\mathscr{G}$ telle que $L = L(\mathscr{G})$ est ambiguë.

Un mot ambigu

Le mot

$$1 + 2 * 3 \in L(\mathscr{G})$$

est ambigu, car :



Un grammaire ambiguë

Soit $\mathscr{G}$

$$\begin{aligned} S &\rightarrow Expr \\ Expr &\rightarrow Expr + Expr \mid Expr * Expr \mid (Expr) \mid Const \\ Const &\rightarrow Chiffre \mid Chiffre\ Const \\ Chiffre &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{aligned}$$

Cette grammaire est ambiguë, car le mot

$$1 + 2 * 3 \in L(\mathscr{G})$$

est ambigu.

Des langages (non) ambigus

Le langage $L(\mathscr{G})$ est non ambigu. Soit $\mathscr{G}'$:

$$\begin{aligned} S &\rightarrow Expr \\ Expr &\rightarrow Expr + Terme \mid Terme \\ Terme &\rightarrow Terme * Facteur \mid Facteur \\ Facteur &\rightarrow (Expr) \mid Const \\ Const &\rightarrow Chiffre \mid Chiffre\ Const \\ Chiffre &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \end{aligned}$$

alors $L(\mathscr{G}') = L(\mathscr{G})$.

Le langage

$$\{ a^n b^n c^m d^m \mid n, m \geq 1 \} \cup \{ a^n b^m c^m d^n \mid n, m \geq 1 \}$$

est ambigu.

Automate à pile

...est un tuplet

$$\langle Q, \Sigma, i, F, \Gamma, \Delta \rangle$$

où

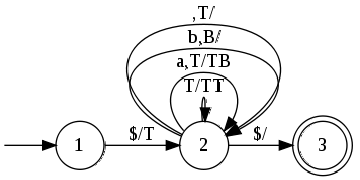
- Q ensemble fini d'états,
- Σ alphabet d'entrée,
- $i \in Q$ état initial, $F \subseteq Q$,
- Γ alphabet de pile,
- Δ, la fonction de transition a le type suivant :

$$\Delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow \mathscr{P}(\Gamma^* \times Q).$$

c'est-à-dire

$$\Delta(q, a, X) \subseteq \Gamma^* \times Q$$

Représentation graphique



- $Q = \{1,2,3\}$, $i = 1$, $F = \{3\}$,
- $\Sigma = \{a, b\}$,
- $\Gamma = \{\$, T, B\}$,
-

Δ	$(\varepsilon, \$)$	(ε, T)	(a, T)	(b, T)	...
1	$\{(T, 2)\}$	$\emptyset$	$\emptyset$	$\emptyset$	
2	$\{(\varepsilon, 3)\}$	$\{(TT, 2), (\varepsilon, 2)\}$	$\{(TB, 2)\}$	$\{(\varepsilon, 3)\}$	
3	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	

Reconnaissance par AP

- Configuration : (q, w, γ) où
  - $q \in Q$,
  - $w \in \Sigma^*$ (la chaîne à lire),
  - et $\gamma \in \Gamma^*$ (la pile).
- Transition :

$$(q, w, \gamma) \vdash (q', w', \gamma') \text{ ssi}$$
$$w = aw',\ \gamma = T\alpha,$$
$$(\beta, q') \in \Delta(q, a, T) \text{ et } \gamma' = \beta\alpha$$
ou
$$w = u,\ \gamma = T\alpha,$$
$$(\beta, q') \in \Delta(q, \varepsilon, T) \text{ et } \gamma' = \beta\alpha.$$

Langage d'un AP

Le mot $w \in L(\mathscr{A})$ ssi
il existe $n \geq 0$ et $(q_0, w_0, \gamma_0), \ldots, (q_n, w_n, \gamma_n)$ tels que :

- $(i, w, \$) = (q_0, w_0, \gamma_0)$,
- $(q_n, w_n, \gamma_n) = (q_n, \varepsilon, \gamma_n)$ et $q_n \in F$,
- $(q_i, w_i, \gamma_i) \vdash (q_{i+1}, w_{i+1}, \gamma_{i+1})$ pour $i = 0, \ldots, n-1$.

Un calcul par AP

Exemple : $abaabb \in L(\mathscr{A})$ car

$$\begin{aligned}(1, abaabb, \$) &\vdash (2, abaabb, T\$)\\ &\vdash (2, abaabb, TT\$)\\ &\vdash (2, baabb, TBT\$)\\ &\vdash (2, baabb, BT\$)\\ &\vdash (2, aabb, T\$)\\ &\vdash (2, abb, TB\$)\\ &\vdash (2, bb, TBB\$)\\ &\vdash (2, bb, BB\$)\\ &\vdash (2, b, B\$)\\ &\vdash (2, , \$)\\ &\vdash (3, ,)\end{aligned}$$

Grammaires, APs, AFNs

Théorème.
Pour tout grammaire $\mathscr{G}$ il existe un AP $\mathscr{A}$ tel que $L(\mathscr{A}) = L(\mathscr{G})$.
Vice-versa :
Pour tout AP $\mathscr{A}$ il existe une grammaire $\mathscr{G}$ tel que $L(\mathscr{G}) = L(\mathscr{A})$.

Théorème.
Pour tout AFN $\mathscr{A}$ il existe une grammaire $\mathscr{G}$ tel que $L(\mathscr{G}) = L(\mathscr{A})$.

Considérez le langage

$$L = \{ a^n b^n \mid n \geq 0 \}.$$

Théorème.
Il existe un AP $\mathscr{A}$ telle que $L = L(\mathscr{A})$.
Il n'existe aucun AFN $\mathscr{A}$ tel que $L = L(\mathscr{A})$.

Le problème $w \in L(\mathscr{G})$

- Algorithme en temps $O(n^3)$:
  - T. Kasami.
    An efficient recognition algorithm for context-free languages.
    Technical Report AFCRL-65-758, Air Force Cambridge research Lab., Bedford, MA, 1965.
  - Daniel H. Younger.
    Recognition and parsing of context-free languages in time n^3.
    Information and Control, 10(2) :189–208, 1967.
- Pas satisfaisant au fin de l'analyse syntaxique.

Prochain objectifs :

- étudier des algorithmes – qui marchent en temps $O(n)$ – adaptés à des sous-classes de grammaires.