LIF15 – Théorie des langages formels

Sylvain Brandel
2019 – 2020
sylvain.brandel@univ-lyon1.fr

CM 10

ANALYSE SYNTAXIQUE

Source : Xavier Urbain

Automates finis

- Transducteurs rationnels
  - Automates finis avec transitions étiquetées par une sorte

    ➔ génération de tokens

- Erreurs jusqu'ici LEXICALES

Grammaires

- Type 0
  - Grammaires générales Pas de restrictions
- Type 1
  - Grammaires contextuelles $uAv \rightarrow uwv$
  - Grammaires croissantes Si $w \rightarrow w' \in R$ alors $|w| \leq |w'|$
- Type 2
  - Grammaires hors contexte Si $w \rightarrow w' \in R$ alors $w \in V$
- Type 3
  - Grammaires régulières

Grammaires

- Grammaires croissantes (contextuelle)

$S \to ABCS$	$S \to aSTc$	$S \to aSBc$	$S \to aSBC$
$S \to T_c$	$S \to aTc$	$S \to abc$	$S \to aBC$
$CA \to AC$	$cT \to Tc$	$cB \to Bc$	$CB \to HB$
$BA \to AB$	$T \to b$	$bB \to bb$	$HB \to HC$
$CB \to BC$			$HC \to BC$
$CT_c \to T_c c$			$aB \to ab$
$CT_c \to T_b c$			$bB \to bb$
$BT_b \to T_b b$			$bC \to bc$
$BT_b \to T_a b$			$cC \to cc$
$AT_a \to T_a a$			
$T_a \to \varepsilon$			

- Grammaire croissante ➔ grammaire contextuelle

Dérivations

- Expressions arithmétiques sur {+, $\times$, (,), id, cte}   tokens
- $E \rightarrow id \mid cte \mid E + E \mid E \times E \mid (E)$

$$x + 2.5 \times 4 + (y + z)$$

$$
\begin{aligned}
E \quad & \Rightarrow \underline{E} + E \\
& \Rightarrow id + \underline{E} \\
& \Rightarrow id + \underline{E} \times E \\
& \Rightarrow id + cte \times \underline{E} \\
& \Rightarrow id + cte \times \underline{E} + E \\
& \Rightarrow id + cte \times cte + \underline{E} \\
& \Rightarrow id + cte \times cte + (\underline{E}) \\
& \Rightarrow id + cte \times cte + (\underline{E} + E) \\
& \Rightarrow id + cte \times cte + (id + \underline{E}) \\
& \Rightarrow id + cte \times cte + (id + id)
\end{aligned}
$$

- Dérivation : gauche
- Parenthésage implicite : $(x + (2.5 \times (4 + (y + z))))$

Dérivations

- Expressions arithmétiques sur $\{+, \times, (,), id, cte\}$
- $E \rightarrow id \mid cte \mid E + E \mid E \times E \mid (E)$

$$x + 2.5 \times 4 + (y + z)$$

$$\begin{aligned}
E \quad & \Rightarrow E + \underline{E} \\
& \Rightarrow E + \underline{(E)} \\
& \Rightarrow E + (E + \underline{E}) \\
& \Rightarrow E + (\underline{E} + id) \\
& \Rightarrow \underline{E} + (id + id) \\
& \Rightarrow E \times \underline{E} + (id + id) \\
& \Rightarrow \underline{E} \times cte + (id + id) \\
& \Rightarrow E + \underline{E} \times cte + (id + id) \\
& \Rightarrow \underline{E} + cte \times cte + (id + id) \\
& \Rightarrow id + cte \times cte + (id + id)
\end{aligned}$$

- Dérivation droite
- Parenthésage implicite : $(((x + 2.5) \times 4) + (y + z))$

Dérivations

- $(((x + 2.5) \times 4) + (y + z)) \neq (x + (2.5 \times (4 + (y + z))))$

- Dérivation droite avec $((x + (2.5 \times 4)) + (y + z))$?

- G ambiguë si $\exists$ w $\in$ L(G) tq plusieurs dérivations droite pour w.

- Lever l'ambiguïté
  - Pas toujours faisable
  - Cas faciles : 1) Nouveau non terminal par niveau de priorité
    2) Récursif gauche si associatif gauche (resp. droite)

Dérivations

- Expressions arithmétiques sur $\{+, \times, (,), id, cte\}$
- $E \rightarrow id \mid cte \mid E + E \mid E \times E \mid (E)$
- $+ < \times$ + moins prioritaire que $\times$
- $+$ et $\times$: associatifs à gauche $x+y+z = (x+y)+z$

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T \times F \mid F$$
$$F \rightarrow id \mid cte \mid (E)$$

- Unique derivation gauche ou droite
- Un peu plus long et complexe mais univoque

Arbre syntaxique

- Plusieurs dérivations pour un même résultat (permutations, etc.)

  ➔ Représentation invariante

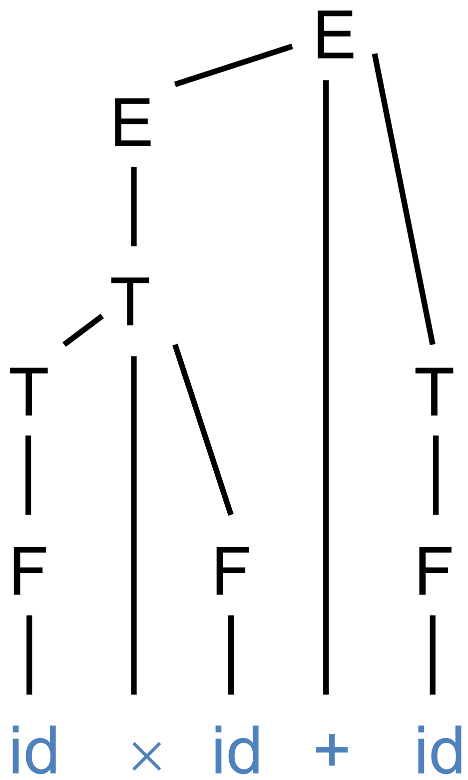
  ➔ Représentation unique lorsque G non ambiguë

- Soit $G = (V, \Sigma, R, S)$, arbres de syntaxe de G :
  - Nœuds internes étiquetés par V
  - Feuilles étiquetées par Σ
  - Si nœud interne N a k fils $a_1, \dots, a_k$ alors $N \rightarrow a_1 \dots a_k \in R$

- Arbre de dérivation : $\Lambda = S$ feuilles $\in \Sigma$

Arbre syntaxique

- Exemple



- Arbre de la dérivation $E \Rightarrow^* \mathrm{id} \times \mathrm{id} + \mathrm{id}$

Arbre syntaxique

- Un arbre de dérivation = plusieurs dérivations
  ➔ Stratégies de parcours (parent avant fils)
- Pour arbre de dérivation : mot des feuilles $\in L(G)$
- Réciproque : produire un arbre, récurrence sur longueur de dérivation
  - Nulle : arbre = feuille a
  - $N \Rightarrow^n w_1 M w_2 \Rightarrow w_1 a_1 \ldots a_k w_2$:
    ajout de k fils au nœud M de l'arbre de $N \Rightarrow^n w_1 M w_2$
- G ambiguë si $\exists\, w \in L(G)$ avec deux arbres de dérivation distincts
- Construction de l'arbre vers le haut ou vers le bas ?

Analyse descendante

- Exemple : grammaire de Dick

  (1) $S \rightarrow (S)S$
  (2) $S \rightarrow \varepsilon$

- $((()())()) \in L(G)$?

- Construction de l'arbre :
  - Lecture à partir de la gauche — Left
  - Construction dérivation gauche — Left
  - Règle déterminée par 1 caractère à produire — 1

  ➔ LL(1)

Analyse descendante

- Efficace
- Simple

- Pas toutes LL(1)
- Si récursif à gauche → échec

- Par exemple $E \rightarrow E + T \mid T$

  $T \rightarrow T \times F \mid F$

  $F \rightarrow id \mid cte \mid (E)$ pas faisable

Analyse ascendante

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T \times F \mid F$$
$$F \rightarrow id \mid cte \mid (E)$$

- Récursive gauche …
- $E \rightarrow T$ ou $E \rightarrow E + T$?
  - ➔ Analyse ascendante
- Construction de l'arbre
  - Lecture à partir de gauche
  - Dérivation droite à l'envers
  - ➔ LR
- En particulier construction d'une forêt

Analyse ascendante

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T \times F \mid F$$
$$F \rightarrow id \mid cte \mid (E)$$

- Racine de forêt : suite des racines des arbres la constituant
- Opérations :
  - 1. Lecture — Shift
  - 2. Enracinement — Reduce

    Sur juxtaposition f f' construction de fN si $N \rightarrow \Lambda(f')$

$id \times id + id$?

Syntaxe abstraite

- Type et signification : depuis l'arbre de syntaxe
- Utilisateur : ce qu'on écrit
- Concrète : presque comme on écrit   Impropre à bonne compréhension

➔ niveau intermédiaire : syntaxe abstraite dépolluée

- On veut :
  - Objectif 1 : sans ambiguïté
  - Objectif 2 : sans scories
  - Objectif 3 : structure et valeurs

Syntaxe abstraite

- T ensemble d'étiquettes abstraites
- $\Sigma = T \cup \{(;); ,\}$
- $G = (V, \Sigma, R, S)$ abstraite si

  1. Règles : $N \to t$

     ou $N \to t(N_1, \ldots, N_k)$ pour $N_i \in V$, $t \in T$
  2. Occurrence t unique dans G

- Mot généré : expression abstraite

$T = \{p, m, cte\}$ $\quad E \to p(E,E) \mid m(E,E) \mid cte$

$T = \{+, \times, cte\}$ $\quad E \to +(E,E) \mid \times(E,E) \mid cte$

Syntaxe abstraite

- G abstraite nécessairement non ambiguë   Objectif 1   OK
- Représentation arborescente :
  - facile grâce à la forme des règles

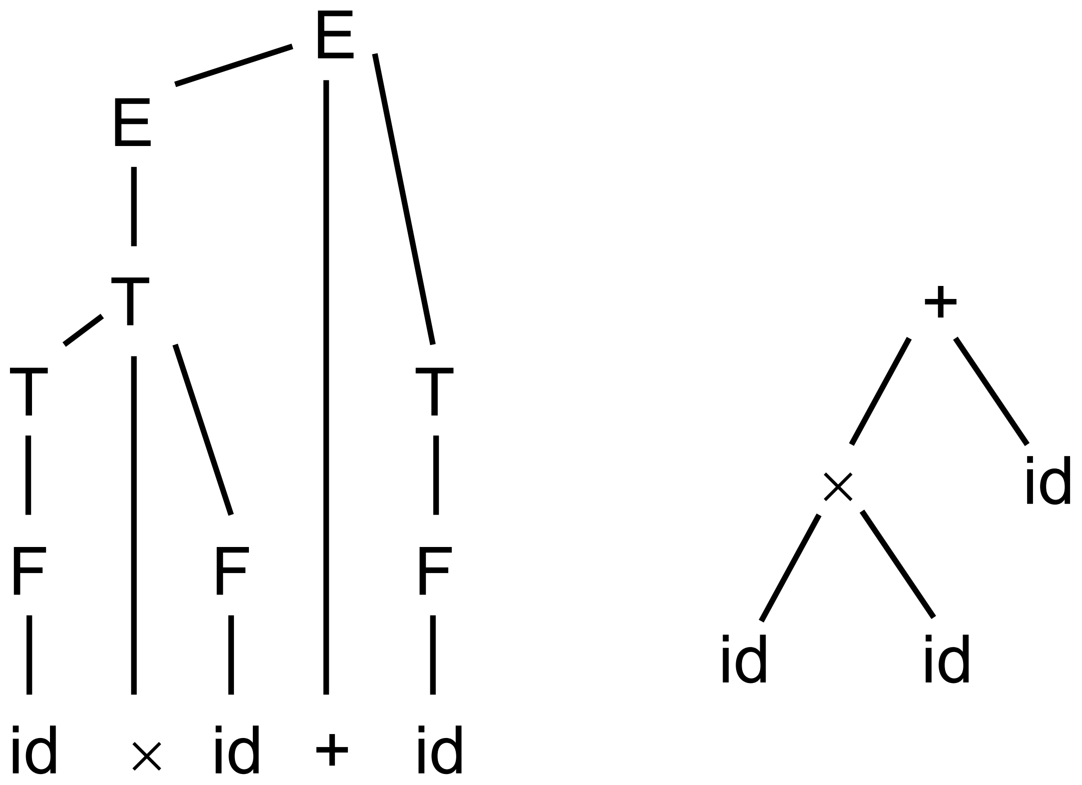
$$\begin{array}{ccc} & t & \\ / & & \backslash \\ a_1 & \ldots & a_k \end{array}$$

- Arbre de syntaxe abstraite A de sorte N : nœuds dans T et
  - $N \to t \in R$ et $A = t$

    ou
  - $N \to t(N_1, \ldots, N_k) \in R$ et $\Lambda(A) = t$
- A a alors k fils ASA de sortes respectives $N_1 \ldots N_k$

Syntaxe abstraite

- Comparaison AS / ASA



- Objectif 2 OK

Syntaxe abstraite

- Pour valeurs : étiquette = langage rationnel

$$E \rightarrow +(E,E) \mid \times (E,E) \mid cte(Nat)$$
$$Nat \in (0|1| \cdots |9)+$$

- Objectif 3 OK

Syntaxe abstraite

- Actions dans une grammaire

$E \rightarrow E + T$	$\{ +(E_1, T_1) \}$
$E \rightarrow T$	$\{ T_1 \}$
$T \rightarrow T \times F$	$\{ \times(T_1, F_1) \}$
$T \rightarrow F$	$\{ F_1 \}$
$F \rightarrow Nat$	$\{ cte(val(Nat_1)) \}$
$F \rightarrow (E)$	$\{ E_1 \}$

- Erreurs ici SYNTAXIQUES