# Analyse syntaxique

Fabienne Carrier & Laurent Mounier & Catherine Parent

**Université Grenoble Alpes**

1er octobre 2019

# Plan

# Analyse syntaxique

Etant donnée une phrase, est-elle conforme à une grammaire donnée ?

Trouver une dérivation

Plusieurs techniques

- Analyse descendante
- Analyse ascendante

La grammaire ne doit pas être ambigüe (un seul arbre pour une phrase donnée)

# Analyse descendante

Analyse deterministe consistant à trouver la dérivation gauche

i.e. dans une dérivation on va toujours développer le non-terminal le plus à gauche

Correspond à un parcours de l'arbre syntaxique de la racine vers les feuilles

# Analyse ascendante

Parcours de l'arbre des feuilles vers la racine

Correspond à une dérivation droite

On n'en verra que l'intuition dans ce cours

# Technique d'analyse descendante

La technique d'analyse déterministe descendante requiert de la grammaire que à chaque choix de règle pour une dérivation, le choix soit dicté par le symbole en cours de lecture dans la chaine d'entŕee

Cette classe de grammaire est appelée : LL(1)

L : parcours de la chaine d'entrée de gauche à droite
L : dérivation gauche
1 : lecture d'un symbole au plus dans la chaîne d'entrée pour choisir une règle

# Analyse descendante : intuition (1/6)

Grammaire G1 :

| | |
|---|---|
| $S \to a\ X\ c$ | $abbdc \in L(G1)$ ? |
| $S \to b\ Y$ | Trouver une règle qui commence par $a$ : |
| $X \to b\ X$ | $S \to a\ X\ c$ |
| $X \to d$ | Puis produire $b$ à partir de $X$ : $X \to b\ X$ |
| $Y \to e$ | Et encore un $b$ : $X \to b\ X$ |
| $Y \to b\ Y$ | Et enfin $d$ : $X \to d$ |
| $Y \to c\ Y\ X$ | On a gagné ! ! ! |

# Analyse descendante : intuition (2/6)

$S \to a\ X\ c$
$S \to b\ Y$
$X \to b\ X$
$X \to d$
$Y \to e$
$Y \to b\ Y$
$Y \to c\ Y\ X$

$bbae \in L(G1)$ ?
Trouver une règle qui commence par $b$ :
$S \to b\ Y$
Puis produire $b$ à partir de $Y$ : $Y \to b\ Y$
Puis un $a$ à partir de $Y$ : pas possible
$bbae \notin L(G1)$

# Analyse descendante : intuition (3/6)

Grammaire G2 :

| | $abd \in L(G2)$ ? |
|---|---|
| $S \to aX$ | $S \to aX \to a\,b\,Y$ |
| $X \to bY$ | ou bien |
| $X \to bZ$ | $S \to aX \to a\,b\,Z$ |
| $Y \to c$ | Laquelle choisir ? |
| $Z \to d$ | Pas de stratégie déterministe |
| | La grammaire n'est pas LL(1) |

Noter que $abd \in L(G2)$ puisqu'il existe une dérivation :
$S \to aX \to a\,b\,Z \to a\,b\,d$
mais pas avec une stratégie déterministe LL(1)

# Analyse descendante : intuition (4/6)

$S \to aX$
$X \to bY$
$X \to bZ$
$Y \to c$
$Z \to d$

Cette grammaire peut être écrite autrement

$S \to a\ b\ U$
$U \to c$
$U \to d$

cette nouvelle grammaire qui décrit le même langage est LL(1)

# Analyse descendante : intuition (5/6)

Grammaire G3 :

$S \to X\ a\ Y$
$X \to W$
$X \to T$
$W \to b\ c$
$T \to a\ c$
$Y \to e\ Y$
$Y \to f$
$Y \to \varepsilon$

$acaef \in L(G3)$

$S \to X\ a\ Y \to W\ a\ Y \to b\ c\ a\ Y \leftarrow$ NON
$S \to X\ a\ Y \to T\ a\ Y \to a\ c\ a\ Y \leftarrow$ OK

Problème ici : pour choisir la bonne règle, on a besoin de connaître les symboles terminaux ($\in V_T$) qui peuvent dériver de la partie droite de la règle (c.a.d. de $W$ ou de $T$)

# Analyse descendante : intuition (6/6)

$S \to X\ a\ Y$
$X \to W$
$X \to T$
$W \to b\ c$
$T \to a\ c$
$Y \to e\ Y$
$Y \to f$
$Y \to \varepsilon$

$S \to X\ a\ Y$
$X \to W \quad \{b\}$
$X \to T \quad \{a\}$
$W \to b\ c$
$T \to a\ c$
$Y \to e\ Y$
$Y \to f$

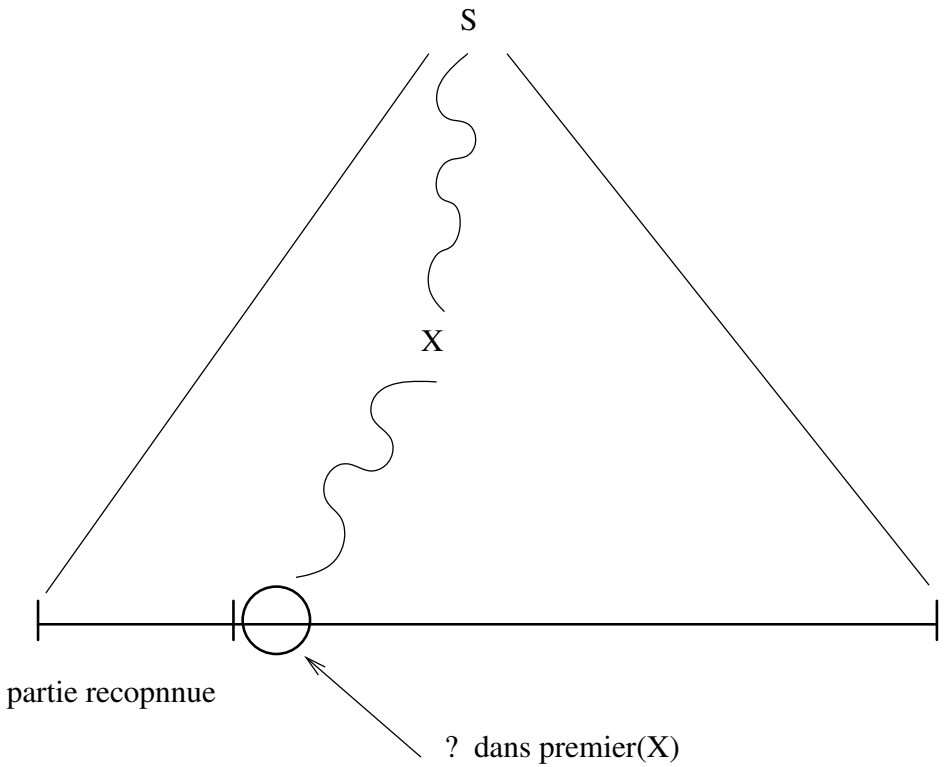
$acaef \in L(G3)$

$S \to X\ a\ Y \to T\ a\ Y \to$
$a\ c\ a\ Y \to a\ c\ a\ e\ Y \to$
$a\ c\ a\ e\ f$

$a$ et $b$ sont appelés symboles directeurs des règles $X \to W$ et $X \to T$

# LL(1) : notion de premier (1/2)

Pour chaque symbole non terminal ($X \in V_n$) on a besoin de connaître quel symbole terminal ($\in V_t$) le plus à gauche peut en dériver.



# LL(1) : notion de premier (2/2)

Si la grammaire comporte deux règles dérivant du même non terminal

$X \to \alpha \qquad \alpha \in (V_t \cup V_n)^*$

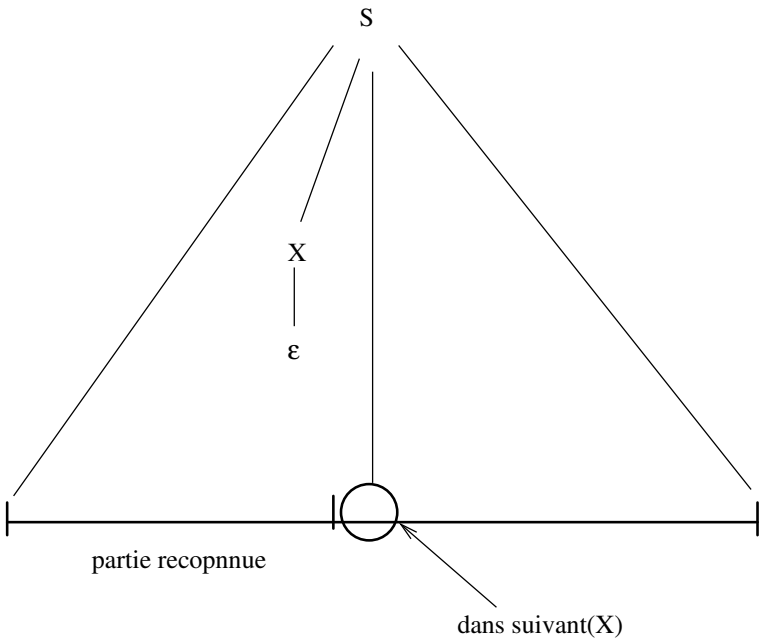
$X \to \beta \qquad \beta \in (V_t \cup V_n)^*$

Propriété LL(1) : $\text{premier}(\alpha) \cap \text{premier}(\beta) = \emptyset$

MAIS on peut avoir des règles de la forme : $X \to \varepsilon$

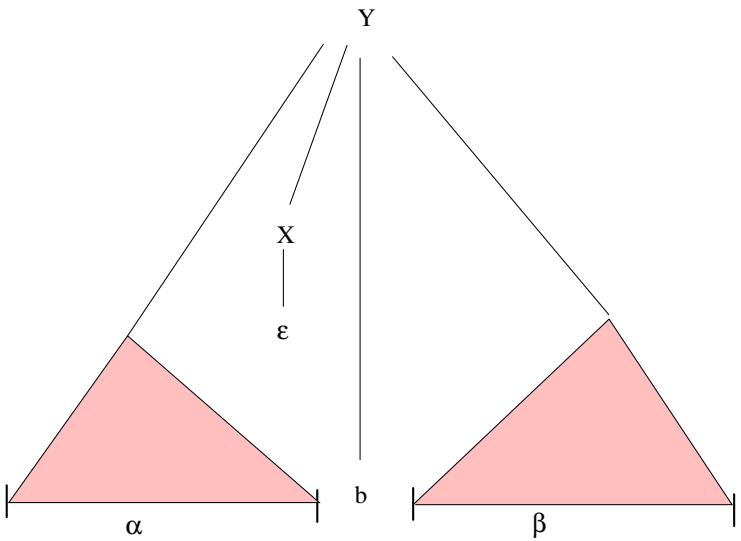
# LL(1) : notion de suivant

Pour $X \in V_n$, suivant($X$) est le symbole qui peut suivre $X$ dans une dérivation.



# LL(1) : suivant, exemple simple

$Y \rightarrow \alpha\, X\, b\, \beta$
$X \rightarrow \varepsilon$




$b \in \text{suivant}(X)$

# LL(1) : définitions Premier, Suivant

$G = (V_t, V_n, Z, P)$

soit $\alpha \in (V_t \cup V_n)*$, **Premiers**($\alpha$) est l'ensemble des symboles terminaux qui peuvent débuter une dérivation de $\alpha$ par $G$ :

$$\textbf{Premier}(\alpha) = \{a \in V_t \mid \alpha \Rightarrow_G^* a\,\beta\}$$

soit $X \in V_n$, **Suivants**($X$) est l'ensemble des symboles terminaux qui peuvent suivre une occurrence de $X$ dans une dérivation de $Z$ par $G$ :

$$\textbf{Suivants}(X) = \{a \in V_t \mid Z \Rightarrow_G^* \alpha\, X\, a\,\beta\}$$

# LL(1) : définition de Vide, Directeur

soit $\alpha \in (V_t \cup V_n)*$, **Vide**$(\alpha)$ vaut vrai ssi $\varepsilon$ peut être dérivé de $\alpha$ par $G$ : **Vide**$(\alpha) \equiv \alpha \Rightarrow_G^* \varepsilon$

soit $X \to u_1.u_2 \ldots u_n$ une règle de $P$, avec $u_i \in (V_t \cup V_n)$

**Directeurs**$(X \to u_1.u_2 \ldots u_n)$ est l'ensemble des symboles terminaux qui peuvent débuter une dérivation par cette règle :

**Directeurs** $(X \to u_1.u_2 \ldots u_n) =$
**Premiers**$(u_1)$
$\cup$ (si **Vide**$(u_1)$ alors **Premiers**$(u_2)$ sinon $\emptyset$)
...
$\cup$ (si **Vide**$(u_1.u_2 \ldots u_{i-1})$ alors **Premiers**$(u_i)$ sinon $\emptyset$)
...
$\cup$ (si **Vide**$(u_1 \ldots u_n)$ alors **Suivants**$(X)$ sinon $\emptyset$)

# Propriété LL(1)

Une grammaire hors-contexte $G = (V_t, V_n, Z, P)$ est dite *LL(1)* si et seulement si l'ensemble des règles de $P$ définissant un même symbole non-terminal ont des directeurs disjoints :

$\forall X \in V_n,\ \forall X \to \alpha,\ X \to \beta \in P$
$\textbf{Directeur}(X \to \alpha) \cap \textbf{Directeur}(X \to \beta) = \emptyset$

Un langage hors-contexte est dit LL(1) si et seulement si il existe une grammaire LL(1) qui le reconnait.
L'ensemble des langages LL(1) est *strictement inclus* dans l'ensemble des langages hors-contexte.

# Exemple 1 : $L1 = a^n b^n c, n \geq 1$ (1/3)

Soit la grammaire $G_1$ qui reconnaît le langage $L_1$

$Z \rightarrow X\ Y\ \sharp$
$X \rightarrow a\ b$
$X \rightarrow a\ X\ b$
$Y \rightarrow c$

Premier($Z$) = Premier($X$)
Premier($X$) = $\{a\}$
Premier($Y$) = $\{c\}$

Suivant($Z$) = $\emptyset$
Suivant($X$) = $\{b\} \cup$ Premier($Y$)
Suivant($Y$) = $\sharp$

# Exemple 1 : $L1 = a^n b^n c, n \geq 1$ (2/3)

Premier($Z$) = Premier($X$)
Premier($X$) = $\{a\}$
Premier($Y$) = $\{c\}$

Suivant($Z$) = $\emptyset$
Suivant($X$) = $\{b\} \cup$ Premier($Y$)
Suivant($Y$) = $\sharp$

Directeurs($Z \rightarrow X\ Y\ \sharp$) = $\{a\}$
Directeurs($X \rightarrow a\ b$) = $\{a\}$
Directeurs($X \rightarrow a\ X\ b$) = $\{a\}$
Directeurs($Y \rightarrow c$) = $\{c\}$

La grammaire G1 n'est donc pas LL(1).

Pour essayer de la rendre LL(1) nous pouvons factoriser la règle dérivant de $X$

# Exemple 1 : $L1 = a^n b^n c, n \geq 1$ (3/3)

Soit la grammaire $G'_1$ qui reconnaît le même langage $L_1$

$Z \rightarrow X\ Y\ \sharp$
$X \rightarrow a\ T$
$T \rightarrow b$
$T \rightarrow X\ b$
$Y \rightarrow c$

Directeurs$(Z \rightarrow X\ Y\ \sharp) = \{a\}$
Directeurs$(X \rightarrow a\ T) = \{a\}$
Directeurs$(T \rightarrow b) = \{b\}$
Directeurs$(T \rightarrow X\ b) = \{a\}$
Directeurs$(Y \rightarrow c) = \{c\}$

La grammaire $G'_1$ est LL(1).

# Exemple 2 : $L2 = a^n b^n c, n \geq 0$

Soit la grammaire $G_2$ qui reconnaît le langage $L_2$

$Z \rightarrow X\ Y\ \sharp$
$X \rightarrow \varepsilon$
$X \rightarrow a\ X\ b$
$Y \rightarrow c$

Directeurs($Z \rightarrow X\ Y\ \sharp$) = Premier(X) = $\{a\} \cup$ Suivant($X$)
Directeurs($X \rightarrow \varepsilon$) = Suivant($X$) = Premier($Y$) $\cup \{b\} = \{b, c\}$
Directeurs($X \rightarrow a\ X\ b$) = $\{a\}$
Directeurs($Y \rightarrow c$) = $\{c\}$

La grammaire $G_2$ est LL(1)

# Exemple 3 : $L3 = ba^n, n \geq 0$

Soit la grammaire $G_3$ qui reconnaît le langage $L_3$

$Z \rightarrow S \sharp$
$S \rightarrow b$
$S \rightarrow S\ a$

Directeurs($S \rightarrow b$) = $\{b\}$
Directeurs($S \rightarrow S\ a$) = Premier($S$) or $b \in$ Premier$(S)$

Cette grammaire n'est pas LL(1)

Toute grammaire récursive à gauche n'est pas LL(1)

# Elimination de la récursivité à gauche (1/2)

Soit la grammaire $G_3$

$Z \rightarrow S \sharp$

$S \rightarrow b$

$S \rightarrow S\ a$

On obtient $G'_3$ qui reconnaît le même langage par élimination de la récursivité à gauche

$Z \rightarrow S \sharp$

$S \rightarrow b\ T$

$T \rightarrow \varepsilon$

$T \rightarrow a\ T$

# Elimination de la récursivité à gauche (2/2)

$Z \rightarrow S\,\sharp$
$S \rightarrow b\,T$
$T \rightarrow \varepsilon$
$T \rightarrow a\,T$

Directeurs($Z \rightarrow S\,\sharp$) = $\{b\}$
Directeurs($S \rightarrow b\,T$ = $\{b\}$
Directeurs($T \rightarrow \varepsilon$) = Suivant($T$) = Suivant($S$) = $\{\sharp\}$
Directeurs($T \rightarrow a\,T$) = $\{a\}$

G'$_3$ est une grammaire LL(1)

# Exemple L4 : $\{b\ a^n, n \geq 1\}$ (1/2)

Soit la grammaire $G_4$

$Z \rightarrow S\ a\ \sharp$

$S \rightarrow b$

$S \rightarrow S\ a$

Après élimination de la récursivité à gauche, soit la grammaire $G'_4$

$Z \rightarrow S\ a\ \sharp$ $\quad$ $T \rightarrow \varepsilon$

$S \rightarrow b\ T$ $\quad$ $T \rightarrow a\ T$

Directeurs($Z \rightarrow S\ a\ \sharp$) = $\{b\}$

Directeurs($S \rightarrow b\ T$ = $\{b\}$

Directeurs($T \rightarrow \varepsilon$) = Suivant($T$) = Suivant($S$) = $\{a\}$

Directeurs($T \rightarrow a\ T$) = $\{a\}$

$G'_4$ n'est pas LL(1)

# Exemple L4 : $\{b\, a^n, n \geq 1\}$ (2/2)

Eliminer la récursivité à gauche ne permet pas toujours de produire une grammaire LL(1) mais nous pouvons écrire la grammaire autrement

$Z \rightarrow b\, A\, \sharp$

$A \rightarrow a\, A$

$A \rightarrow a$

qui une fois factorisée donne

$Z \rightarrow b\, A\, \sharp$

$A \rightarrow a\, X$

$X \rightarrow \varepsilon$

$X \rightarrow A$

Directeurs($X \rightarrow \varepsilon$) = Suivant($X$) = Suivant($A$) = $\{\sharp\}$

Directeurs($X \rightarrow A$) = Premier($A$) = $\{a\}$

# Grammaire non LL($k$) $\forall k$

Le langage des palindrômes ne peut pas être décrit par une grammaire LL(k)

Considérons le vocabulaire terminal $\{a, b\}$

$Z \rightarrow S \sharp$
$S \rightarrow a\ S\ a$
$S \rightarrow b\ S\ b$
$S \rightarrow \varepsilon$

Directeurs($S \rightarrow a\ S\ a$) = $\{a\}$
Directeurs($S \rightarrow b\ S\ b$) = $\{b\}$
Directeurs($S \rightarrow \varepsilon$) = $\{a, b, \sharp\}$

# Construction d'un reconnaisseur (1/3)

1- $Z \rightarrow S\ \sharp$
2- $S \rightarrow A\ C$
3- $A \rightarrow a\ A\ b$
4- $A \rightarrow \varepsilon$
5- $C \rightarrow c\ Y$
6- $Y \rightarrow c\ Y$
7- $Y \rightarrow \varepsilon$

$\text{Directeurs}(Z \rightarrow S\ \sharp) = \{a, b, c\}$
$\text{Directeurs}(S \rightarrow A\ C) = \{a, b, c\}$
$\text{Directeurs}(A \rightarrow a\ A\ b) = \{a\}$
$\text{Directeurs}(A \rightarrow \varepsilon) = \{b, c\}$
$\text{Directeurs}(C \rightarrow c\ Y) = \{c\}$
$\text{Directeurs}(Y \rightarrow c\ Y) = \{c\}$
$\text{Directeurs}(Y \rightarrow \varepsilon) = \{\sharp\}$

# Construction d'un reconnaisseur (2/3)

## Table de prédiction

| | a | b | c | ♯ |
|---|---|---|---|---|
| Z | 1 | 1 | 1 | |
| S | 2 | 2 | 2 | |
| A | 3 | 4 | 4 | |
| C | | | 5 | |
| Y | | | 6 | 7 |

| entrée | pile |
|---|---|
| aabbc♯ | Z |
| aabbc♯ | S ♯ |
| aabbc♯ | A C ♯ |
| aabbc♯ | a A b C ♯ |
| abbc♯ | A b C ♯ |
| abbc♯ | a A b b C ♯ |
| bbc♯ | A b b C ♯ |
| bbc♯ | b b C ♯ |
| bc♯ | b C ♯ |
| c♯ | C ♯ |
| c♯ | c Y♯ |
| ♯ | Y♯ |
| ♯ | ♯ |

# Construction d'un reconnaisseur (3/3)

Programmation systématique avec procédures récursives

```
reconnaitreS :
(* S --> A C *)
   si symbole dans {a, b, c}
   alors reconnaitreA ; reconnaitreC


reconnaitreA :
(* A --> a A b *)
   si symbole = a
   alors lire; reconnaitreA; si symbole = b alors lire
   sinon si symbole dans {b, c}
         alors rien
         sinon erreur
```