

Langages et Compilation

Analyse descendante prédictive

Introduction

Grammaires $LL(1)$

Une famille de grammaires analysables de façon efficace.

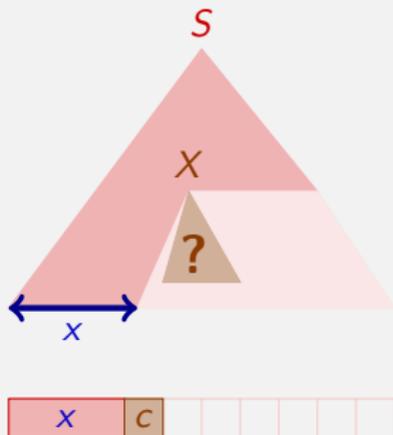
Caractéristiques de l'analyse $LL(1)$

- **analyse descendante**

Construction de l'arbre de dérivation selon l'ordre préfixé :
on part de la racine (l'axiome) et on descend vers les feuilles (les terminaux) en développant le nœud interne (une variable) le plus à gauche.

- **analyse prédictive**

Pour développer le nœud, on choisit la règle à appliquer en prévisualisant le symbole courant du mot analysé.



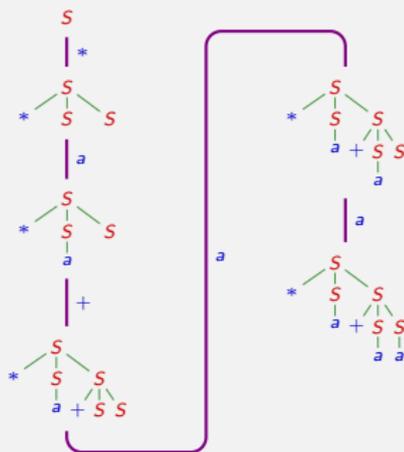
Introduction

Exemple $S \rightarrow +SS \mid *SS \mid a$

Les chaînes dérivées de S commencent toutes par un terminal distinct
 \rightsquigarrow À partir du terminal courant du mot analysé, on sait déterminer la bonne règle.

	$+$	$*$	a
S	$S \rightarrow +SS$	$S \rightarrow *SS$	$S \rightarrow a$

Analyse du mot $*a + aa$



Introduction

Exemple $S \rightarrow aSbS \mid \varepsilon$

Le choix devient plus délicat lorsque la grammaire comprend des ε -productions. Quel critère doit-on prendre en compte pour choisir entre la règle $S \rightarrow aSbS$ et la règle $S \rightarrow \varepsilon$?

On est amené à considérer les terminaux qui peuvent suivre S .

Convention pratique, on introduit un terminal particulier $\$$ pour marquer la fin du mot à analyser.

$\$$ et b (mais pas a) peuvent se retrouver à droite de S .

\leadsto On sait alors déterminer la règle à appliquer au vu du terminal courant.

	$\$$	a	b
S	$S \rightarrow \varepsilon$	$S \rightarrow aSbS$	$S \rightarrow \varepsilon$

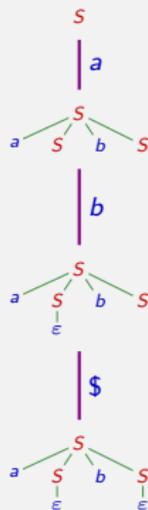
Introduction

Exemple $S \rightarrow aSbS \mid \epsilon$

sa table d'analyse

	\$	a	b
S	$S \rightarrow \epsilon$	$S \rightarrow aSbS$	$S \rightarrow \epsilon$

Analyse du mot $ab\$$



Condition pour une analyse $LL(1)$ à chaque étape, pour la variable X à développer et le terminal courant c en entrée, le choix de la dérivation à appliquer doit être déterministe.

- L** *lecture de l'entrée de la gauche vers la droite (Left to right scanning)*
- L** *construction d'une dérivation gauche (Left derivation)*
- 1** *symbole de l'entrée pour prédire la bonne règle*

L'analyse $LL(1)$ se base sur une table qui indique, pour la variable X et le terminal c , la règle correcte à appliquer.

Pour construire cette table d'analyse, on détermine au préalable :

1. les variables effaçables
2. les ensembles Premier
3. les ensembles Suivant

Construction de la table d'analyse

Grammaire $LL(1)$

Analyseur $LL(1)$

Rendre une grammaire $LL(1)$

Analyse $LL(k)$

Les variables effaçables

Une chaîne $\alpha \in N^*$ est dite **effaçable** si le mot vide se dérive de α :
 $\alpha \xrightarrow{*} \varepsilon$

Calcul des variables effaçables

On construit par récurrence sur i , l'ensemble Eff_i des variables A .

$$\text{Eff}_0 = \{A \in N : A \rightarrow \varepsilon \in P\}$$

$$\text{Eff}_{i+1} = \{A \in N : A \rightarrow \alpha \in P \text{ et } \alpha \in \text{Eff}_i^*\}$$

Arrêt lorsque $\text{Eff}_{i+1} = \text{Eff}_i$ (au bout d'au plus $|N|$ étapes)

Exemple

$$\left\{ \begin{array}{l} S \rightarrow XY \\ X \rightarrow aXb \mid \varepsilon \\ Y \rightarrow cZ \mid Ze \\ Z \rightarrow dcZ \mid \varepsilon \end{array} \right.$$

$$\text{Eff}_0 = \{X, Z\} = \text{Eff}_1$$

Les ensembles Premier

Soit α une chaîne de $(\Sigma \cup N)^*$. $\text{Premier}(\alpha)$ est l'ensemble des terminaux qui débutent les chaînes dérivées de α :

$$\left\{ a \in \Sigma : \alpha \xrightarrow{*} a\beta \text{ où } \beta \in (\Sigma \cup N)^* \right\}$$

Calcul des ensembles Premier

Pour un terminal a :

$$\text{Premier}(a\beta) = \{a\}$$

Pour une variable X :

$$\text{Premier}(X) = \bigcup_{X \rightarrow \beta \in P} \text{Premier}(\beta)$$

$$\text{Premier}(X\beta) =$$

$$\begin{cases} \text{Premier}(X) & \text{si } X \text{ n'est pas effaçable} \\ \text{Premier}(X) \cup \text{Premier}(\beta) & \text{sinon} \end{cases}$$

Les ensembles Premier

Exemple

$$\left\{ \begin{array}{l} S \rightarrow XY \\ X \rightarrow aXb \mid \varepsilon \\ Y \rightarrow cZ \mid Ze \\ Z \rightarrow dcZ \mid \varepsilon \end{array} \right.$$

Calcul des ensembles Premier pour les variables :

$$\text{Premier}(S) = \text{Premier}(XY) \quad S \rightarrow XY$$

$$= \text{Premier}(X) \cup \text{Premier}(Y) \quad X \text{ effaçable}$$

$$\text{Premier}(X) = \text{Premier}(aXb) \quad X \rightarrow aXb \mid \varepsilon$$

$$= \{a\}$$

$$\text{Premier}(Y) = \text{Premier}(cZ) \cup \text{Premier}(Ze) \quad Y \rightarrow cZ \mid Ze$$

$$= \{c\} \cup \text{Premier}(Z) \cup \{e\} \quad Z \text{ effaçable}$$

$$\text{Premier}(Z) = \text{Premier}(dcZ) \quad Z \rightarrow dcZ \mid \varepsilon$$

$$= \{d\}$$

Les ensembles Suivant

Soit X une variable. $\text{Suivant}(X)$ est l'ensemble des terminaux qui peuvent apparaître après X dans une dérivation :

$$\left\{ a \in \Sigma : S \xrightarrow{*} \alpha X a \beta \text{ où } \alpha, \beta \in (\Sigma \cup N)^* \right\}$$

i.e.

$$\text{Suivant}(X) = \bigcup_{Y \rightarrow \alpha X \beta \in P} \text{Premier}(\beta \text{ Suivant}(Y))$$

Calcul des ensembles Suivant

Mettre $\$$ dans $\text{Suivant}(S)$ (où S est l'axiome)

Pour chaque variable X , examiner chaque production $Y \rightarrow \alpha X \beta$ où X est à droite :

- Si $\beta \neq \varepsilon$, ajouter les éléments de $\text{Premier}(\beta)$ à $\text{Suivant}(X)$
- Si $\beta \xrightarrow{*} \varepsilon$, ajouter les éléments de $\text{Suivant}(Y)$ à $\text{Suivant}(X)$

Les ensembles Suivant

Exemple

$$\left\{ \begin{array}{l} S \rightarrow XY \\ X \rightarrow aXb \mid \varepsilon \\ Y \rightarrow cZ \mid Ze \\ Z \rightarrow dcZ \mid \varepsilon \end{array} \right.$$

Calcul des ensembles Suivant :

Suivant(S) contient \$

Suivant(X)

$$\begin{aligned} S \rightarrow XY &\rightsquigarrow \text{Premier}(Y \text{ Suivant}(S)) \subseteq \text{Suivant}(X) \\ &\rightsquigarrow \text{Premier}(Y) \subseteq \text{Suivant}(X) \end{aligned}$$

(Y non effaçable)

$$X \rightarrow aXb \rightsquigarrow b \in \text{Suivant}(X)$$

Suivant(Y)

$$S \rightarrow XY \rightsquigarrow \text{Suivant}(S) \subseteq \text{Suivant}(Y)$$

Suivant(Z)

$$Y \rightarrow cZ \rightsquigarrow \text{Suivant}(Y) \subseteq \text{Suivant}(Z)$$

$$Y \rightarrow Ze \rightsquigarrow e \in \text{Suivant}(Z)$$

$$Z \rightarrow dcZ \rightsquigarrow \text{Suivant}(Z) \subseteq \text{Suivant}(Z)$$

Les variables Effaçables et les ensembles Premier et Suivant

Exemple

$$\left\{ \begin{array}{l} S \rightarrow XY \\ X \rightarrow aXb \mid \varepsilon \\ Y \rightarrow cZ \mid Ze \\ Z \rightarrow dcZ \mid \varepsilon \end{array} \right.$$

Bilan des calculs :

	Effaçable	Premier	Suivant
<i>S</i>	non	<i>a c d e</i>	\$
<i>X</i>	oui	<i>a</i>	<i>b c d e</i>
<i>Y</i>	non	<i>c d e</i>	\$
<i>Z</i>	oui	<i>d</i>	\$ <i>e</i>

Construction de la table d'analyse

Table à deux dimensions indexée par les variables et les terminaux.
Les éléments de la table sont les productions de la grammaire.

Pour toute production $X \rightarrow \alpha$:

- Ajouter $X \rightarrow \alpha$ à l'entrée $T[X, a]$ pour tout terminal a dans $\text{Premier}(\alpha)$
- Si α est effaçable, ajouter $X \rightarrow \alpha$ dans la case $T[X, a]$ pour tout terminal a dans $\text{Suivant}(X)$

Construction de la table d'analyse

$$\begin{cases} S \rightarrow XY \\ X \rightarrow aXb \mid \varepsilon \\ Y \rightarrow cZ \mid Ze \\ Z \rightarrow dcZ \mid \varepsilon \end{cases}$$

	Effaçable	Premier	Suivant
S	non	a c d e	\$
X	oui	a	b c d e
Y	non	c d e	\$
Z	oui	d	\$ e

$$S \rightarrow XY : \text{Premier}(XY) = \text{Premier}(X) \cup \text{Premier}(Y) = \{a, c, d, e\}$$

$$X \rightarrow aXb : \text{Premier}(aXb) = \{a\}$$

$$Y \rightarrow cZ : \text{Premier}(cZ) = \{c\}$$

$$Z \rightarrow dcZ : \text{Premier}(dcZ) = \{d\}$$

$$Y \rightarrow Ze : \text{Premier}(Ze) = \text{Premier}(Z) \cup \text{Premier}(e) = \{d, e\}$$

$$X \rightarrow \varepsilon : \text{Suivant}(X) = \{b, c, d, e\}$$

$$Z \rightarrow \varepsilon : \text{Suivant}(Z) = \{\$, e\}$$

	\$	a	b	c	d	e
S		$S \rightarrow XY$		$S \rightarrow XY$	$S \rightarrow XY$	$S \rightarrow XY$
X		$X \rightarrow aXb$	$X \rightarrow \varepsilon$	$X \rightarrow \varepsilon$	$X \rightarrow \varepsilon$	$X \rightarrow \varepsilon$
Y				$Y \rightarrow cZ$	$Y \rightarrow Ze$	$Y \rightarrow Ze$
Z	$Z \rightarrow \varepsilon$				$Z \rightarrow dcZ$	$Z \rightarrow \varepsilon$

Construction de la table d'analyse

Grammaire $LL(1)$

Analyseur $LL(1)$

Rendre une grammaire $LL(1)$

Analyse $LL(k)$

Définition

Une grammaire est $LL(1)$ s'il existe au plus une production par entrée dans la table.

Proposition

Une grammaire est $LL(1)$ si pour toute paire de productions $A \rightarrow \alpha$ et $A \rightarrow \beta$, on a :

$$\text{Premier}(\alpha \text{ Suivant}(A)) \cap \text{Premier}(\beta \text{ Suivant}(A)) = \emptyset$$

Ainsi une grammaire ne sera pas $LL(1)$ s'il on a :

- soit un **conflit Premier/Premier**, i.e., deux règles $A \rightarrow \alpha \mid \beta$ telles que : $\text{Premier}(\alpha) \cap \text{Premier}(\beta) \neq \emptyset$
- soit un **conflit Premier/Suivant**, i.e., deux règles $A \rightarrow \alpha \mid \beta$ avec $\beta \xrightarrow{*} \varepsilon$ telles que : $\text{Premier}(\alpha) \cap \text{Suivant}(A) \neq \emptyset$

Grammaire $LL(1)$

Exemple

$$\left\{ \begin{array}{l} S \rightarrow XY \\ X \rightarrow aXb \mid \varepsilon \\ Y \rightarrow cZ \mid Ze \\ Z \rightarrow dcZ \mid \varepsilon \end{array} \right. \text{ est une grammaire } LL(1)$$

Sa table d'analyse comprend au plus une alternative par case.

~> Le choix de la règle à appliquer se fait de façon déterministe.

	\$	a	b	c	d	e
S		$S \rightarrow XY$		$S \rightarrow XY$	$S \rightarrow XY$	$S \rightarrow XY$
X		$X \rightarrow aXb$	$X \rightarrow \varepsilon$	$X \rightarrow \varepsilon$	$X \rightarrow \varepsilon$	$X \rightarrow \varepsilon$
Y				$Y \rightarrow cZ$	$Y \rightarrow Ze$	$Y \rightarrow Ze$
Z	$Z \rightarrow \varepsilon$				$Z \rightarrow dcZ$	$Z \rightarrow \varepsilon$

Grammaire $LL(1)$

Exemple

$$\left\{ \begin{array}{l} S \rightarrow XY \mid Z \\ X \rightarrow aXb \mid \varepsilon \\ Y \rightarrow bX \\ Z \rightarrow bZc \mid c \end{array} \right. \text{ n'est pas une grammaire } LL(1)$$

Il existe un conflit Premier/Premier pour les règles $S \rightarrow XY$ et $S \rightarrow Z$

$$\text{Premier}(XY) = \text{Premier}(X) \cup \text{Premier}(Y) = \{a, b\}$$

X effaçable

$$\text{Premier}(Z) = \{b, c\}$$

$$\rightsquigarrow \text{Premier}(XY) \cap \text{Premier}(Z) = \{b\} \neq \emptyset$$

	\$	b	...
...			
S		$S \rightarrow XY$ $S \rightarrow Z$	

Grammaire $LL(1)$

Exemple

$$\begin{cases} S \rightarrow aXb \\ X \rightarrow bX \mid \varepsilon \end{cases} \text{ n'est pas une grammaire } LL(1)$$

Il existe un conflit Premier/Suivant pour les règles $X \rightarrow bX$ et $X \rightarrow \varepsilon$

$$\text{Premier}(bX) = \{b\}$$

$$\text{Suivant}(X) = \{b\}$$

$$\rightsquigarrow \text{Premier}(X) \cap \text{Suivant}(X) = \{b\} \neq \emptyset$$

	\$	b	...
...			
X		$X \rightarrow bX$ $X \rightarrow \varepsilon$	

Construction de la table d'analyse

Grammaire $LL(1)$

Analyseur $LL(1)$

Rendre une grammaire $LL(1)$

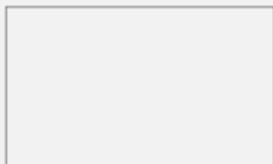
Analyse $LL(k)$

Analyseur $LL(1)$

Pour examiner un mot, l'analyseur $LL(1)$ utilise la table d'analyse précédemment construite et une pile.

Initialement, la pile contient le marqueur de fin de mot \$ et l'axiome.

Le mot analysé lu de gauche à droite



La table d'analyse
de la grammaire $LL(1)$

Analyseur
 $LL(1)$

S

L'arbre d'analyse
à construire



La pile

Analyseur $LL(1)$

À chaque étape, on examine le terminal courant c du mot analysé et le sommet de la pile (premier symbole non traité de l'arbre en construction).

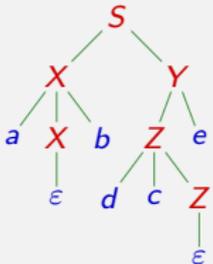
- Soit le sommet de la pile est un terminal a :
 - si $a \neq c$,
l'analyse s'arrête et retourne **ÉCHEC**
 - si $a = c = \$$,
l'analyse s'arrête et retourne **SUCCÈS**
 - si $a = c \neq \$$,
 a est dépilé et on avance dans la lecture du mot analysé
- Soit le sommet de la pile est une variable X :
 - si l'entrée $T[X, c]$ est vide,
l'analyse s'arrête et retourne **ÉCHEC**
 - si l'entrée $T[X, c]$ contient une règle $X \rightarrow \alpha$,
 X est dépilé et α est empilé en partant de la droite
(par exemple, si $X \rightarrow YzT$, T est empilé, puis z , puis Y).

Analyseur LL(1)

	\$	a	b	c	d	e
S		$S \rightarrow XY$		$S \rightarrow XY$	$S \rightarrow XY$	$S \rightarrow XY$
X		$X \rightarrow aXb$	$X \rightarrow \epsilon$	$X \rightarrow \epsilon$	$X \rightarrow \epsilon$	$X \rightarrow \epsilon$
Y				$Y \rightarrow cZ$	$Y \rightarrow Ze$	$Y \rightarrow Ze$
Z	$Z \rightarrow \epsilon$				$Z \rightarrow dcZ$	$Z \rightarrow \epsilon$

Mot analysé	Pile	
<i>abdce</i> \$	<i>S</i> \$	$S \rightarrow XY$ 
<i>abdce</i> \$	<i>XY</i> \$	$X \rightarrow aXb$ 
<i>abdce</i> \$	<i>aXbY</i> \$	assortiment

Mot analysé	Pile	
<i>abdce</i> \$	<i>XbY</i> \$	$X \rightarrow \epsilon$
		<pre> graph TD S --> X1[X] S --> Y1[Y] X1 --> a1[a] X1 --> X2[X] X1 --> b1[b] X2 --> epsilon1[ε] </pre>
<i>abdce</i> \$	<i>bY</i> \$	assortiment
<i>abdce</i> \$	<i>Y</i> \$	$Y \rightarrow Ze$
		<pre> graph TD S --> X1[X] S --> Y1[Y] X1 --> a1[a] X1 --> X2[X] X1 --> b1[b] X2 --> epsilon1[ε] Y1 --> Z1[Z] Y1 --> e1[e] </pre>
<i>abdce</i> \$	<i>Ze</i> \$	$Z \rightarrow dcZ$
		<pre> graph TD S --> X1[X] S --> Y1[Y] X1 --> a1[a] X1 --> X2[X] X1 --> b1[b] X2 --> epsilon1[ε] Y1 --> Z1[Z] Y1 --> e1[e] Z1 --> d1[d] Z1 --> c1[c] Z1 --> Z2[Z] </pre>
<i>abdce</i> \$	<i>dcZe</i> \$	assortiment

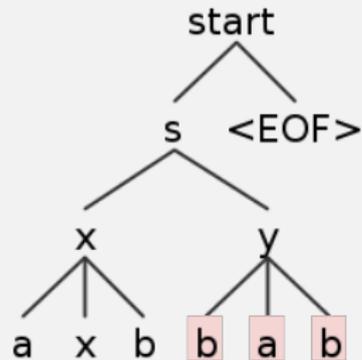
Mot analysé	Pile	
<i>abdce</i> \$	<i>cZe</i> \$	assortiment
<i>abdce</i> \$	<i>Ze</i> \$	$Z \rightarrow \epsilon$
		
<i>abdce</i> \$	<i>e</i> \$	assortiment
<i>abdce</i> \$	<i>\$</i>	SUCCES

Le coût de l'analyse du mot est linéaire en la taille de l'arbre.

Analyseur LL(1)

	\$	a	b	c	d	e
S		$S \rightarrow XY$		$S \rightarrow XY$	$S \rightarrow XY$	$S \rightarrow XY$
X		$X \rightarrow aXb$	$X \rightarrow \epsilon$	$X \rightarrow \epsilon$	$X \rightarrow \epsilon$	$X \rightarrow \epsilon$
Y				$Y \rightarrow cZ$	$Y \rightarrow Ze$	$Y \rightarrow Ze$
Z	$Z \rightarrow \epsilon$				$Z \rightarrow dcZ$	$Z \rightarrow \epsilon$

Mot analysé	Pile	
<i>abbab</i> \$	S\$	$S \rightarrow XY$
<i>abbab</i> \$	XY\$	$X \rightarrow aXb$
<i>abbab</i> \$	aXbY\$	assortiment
<i>abbab</i> \$	XbY\$	$X \rightarrow \epsilon$
<i>abbab</i> \$	bY\$	assortiment
<i>abbab</i> \$	Y\$	ECHEC



Construction de la table d'analyse

Grammaire $LL(1)$

Analyseur $LL(1)$

Rendre une grammaire $LL(1)$

Analyse $LL(k)$

Rendre une grammaire $LL(1)$

Proposition

Une grammaire ne peut pas être $LL(1)$ si elle est :

soit ambiguë,

soit récursive gauche,

soit n'est pas factorisée à gauche.

On peut modifier la grammaire pour tenter de la rendre $LL(1)$ mais le résultat n'est pas garanti.

Proposition

Il existe des grammaires non ambiguës, non récursives gauche et factorisées à gauche qui ne sont pas $LL(1)$.

Rendre une grammaire $LL(1)$

Par élimination de la récursivité gauche

Exemple

$$\left\{ \begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow (E) \mid nb \end{array} \right.$$

n'est pas $LL(1)$ car récursive gauche

Il existe un conflit Premier/Premier pour les règles $E \rightarrow E + T$ et $E \rightarrow T$ à cause de la récursivité gauche de E

et un conflit Premier/Premier pour les règles $T \rightarrow T * F$ et $T \rightarrow F$ à cause de la récursivité gauche de T

	\$	(nb)	+	*
E		$E \rightarrow E + T$ $E \rightarrow T$	$E \rightarrow E + T$ $E \rightarrow T$			
T		$T \rightarrow T * F$ $T \rightarrow F$	$T \rightarrow T * F$ $T \rightarrow F$			
F		$F \rightarrow (E)$	$F \rightarrow nb$			

Rendre une grammaire $LL(1)$

Par élimination de la récursivité gauche

Supprimer la récursivité gauche rend cette grammaire $LL(1)$

$$\left\{ \begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow (E) \mid nb \end{array} \right. \longrightarrow \left\{ \begin{array}{l} E \rightarrow TY \\ Y \rightarrow +TY \mid \varepsilon \\ T \rightarrow FZ \\ Z \rightarrow *FZ \mid \varepsilon \\ F \rightarrow (E) \mid nb \end{array} \right.$$

Rendre une grammaire $LL(1)$

Par élimination de la récursivité gauche

$$\left\{ \begin{array}{l} E \rightarrow TY \\ Y \rightarrow +TY \mid \varepsilon \\ T \rightarrow FZ \\ Z \rightarrow *FZ \mid \varepsilon \\ F \rightarrow (E) \mid nb \end{array} \right.$$

	Effaçable	Premier	Suivant
E	non	(nb	\$)
Y	oui	+	\$)
T	non	(nb	\$) +
Z	oui	*	\$) +
F	non	(nb	* \$) +

	\$	(nb)	+	*
E		$E \rightarrow TY$	$E \rightarrow TY$			
Y	$Y \rightarrow \varepsilon$			$Y \rightarrow \varepsilon$	$Y \rightarrow +TY$	
T		$T \rightarrow FZ$	$T \rightarrow FZ$			
Z	$Z \rightarrow \varepsilon$			$Z \rightarrow \varepsilon$	$Z \rightarrow \varepsilon$	$Z \rightarrow *FZ$
F		$F \rightarrow (E)$	$F \rightarrow nb$			

Rendre une grammaire $LL(1)$

Par substitution et factorisation

Exemple

$$\left\{ \begin{array}{l} E \rightarrow TR \\ T \rightarrow id \mid (E) \mid A \\ R \rightarrow +E \mid \varepsilon \\ A \rightarrow id[E] \end{array} \right. \quad \text{n'est pas } LL(1)$$

Il existe un conflit Premier/Premier pour les règles $T \rightarrow id$ et $T \rightarrow A$

	\$	id	...
...			
T		$T \rightarrow id$ $T \rightarrow A$	

Rendre une grammaire $LL(1)$

Par substitution et factorisation

On effectue une substitution avant de factoriser à gauche.

$$\begin{cases} E \rightarrow TR \\ T \rightarrow id \mid (E) \mid A \\ R \rightarrow +E \mid \varepsilon \\ A \rightarrow id[E] \end{cases}$$

- Substitution $\begin{cases} E \rightarrow TR \\ T \rightarrow id \mid (E) \mid id[E] \\ R \rightarrow +E \mid \varepsilon \end{cases}$

- Factorisation à gauche $\begin{cases} E \rightarrow TR \\ T \rightarrow id X \mid (E) \\ X \rightarrow [E] \mid \varepsilon \\ R \rightarrow +E \mid \varepsilon \end{cases}$

Rendre une grammaire $LL(1)$

Par substitution et factorisation

On obtient une grammaire $LL(1)$

	Effaçable	Premier	Suivant
E	non	$id ($	$\$)]$
T	non	$id ($	$\$)] +$
X	oui	$[$	$\$)] +$
R	oui	$+$	$\$)]$

	$\$$	$+$	id	$($	$)$	$[$	$]$
E			$E \rightarrow TR$	$E \rightarrow TR$			
T			$T \rightarrow id X$	$T \rightarrow (E)$			
X	$X \rightarrow \varepsilon$	$X \rightarrow \varepsilon$		$X \rightarrow \varepsilon$		$X \rightarrow [E]$	$X \rightarrow \varepsilon$
R	$R \rightarrow \varepsilon$	$R \rightarrow +E$			$R \rightarrow \varepsilon$		$R \rightarrow \varepsilon$

Rendre une grammaire $LL(1)$

ou rendre l'analyse déterministe

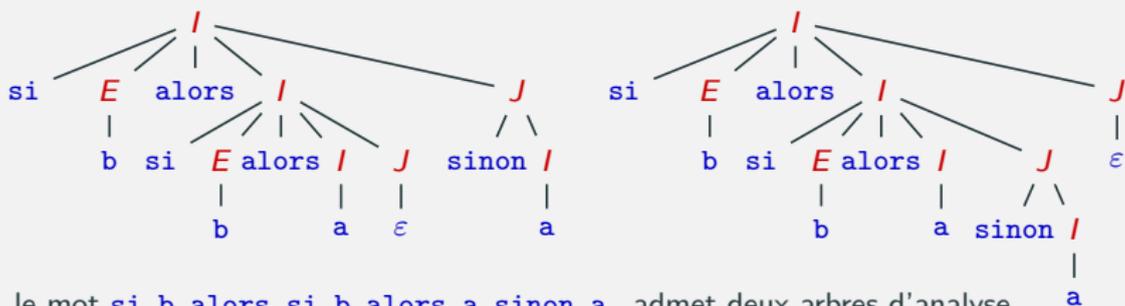
La grammaire des instructions de branchements conditionnels

$$\begin{cases} I \rightarrow \text{si } E \text{ alors } I \text{ sinon } I \mid \text{si } E \text{ alors } I \mid a \\ E \rightarrow b \end{cases}$$

même factorisée

$$\begin{cases} I \rightarrow \text{si } E \text{ alors } I J \mid a \\ J \rightarrow \text{sinon } I \mid \epsilon \\ E \rightarrow b \end{cases}$$

n'est pas $LL(1)$ car ambiguë.



le mot **si b alors si b alors a sinon a** admet deux arbres d'analyse

Rendre une grammaire $LL(1)$

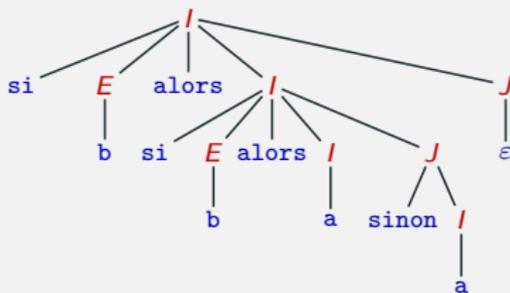
ou rendre l'analyse déterministe

L'ambiguïté engendre un conflit Premier/Suivant pour les règles

$J \rightarrow \text{sinon } I$ et $J \rightarrow \epsilon$

	\$	<i>sinon</i>	...
...			
<i>J</i>	$J \rightarrow \epsilon$	$J \rightarrow \text{sinon } I$ $J \rightarrow \epsilon$...

Avec la convention usuelle d'associer le *sinon* avec le *si* le plus proche, on peut rendre l'analyseur déterministe et le forcer à produire l'arbre voulu en privilégiant la règle $J \rightarrow \text{sinon } I$ au détriment de la règle $J \rightarrow \epsilon$.



Construction de la table d'analyse

Grammaire $LL(1)$

Analyseur $LL(1)$

Rendre une grammaire $LL(1)$

Analyse $LL(k)$

Analyse $LL(k)$

Généralisation de l'analyse $LL(1)$ avec prévisualisation non pas juste d'un symbole mais d'un nombre fixé k de symboles. Une grammaire est $LL(k)$ si l'analyseur peut choisir de façon déterministe la règle à appliquer en examinant les k symboles courants de l'entrée.

On note :

- $w|_k = \begin{cases} w & \text{si } w \text{ est de longueur au plus } k \\ \text{le préfixe de longueur } k \text{ de } w & \text{sinon} \end{cases}$
- $\text{Premier}_k(\alpha) = \{w|_k : \alpha \xrightarrow{*} w\}$
- $\text{Suivant}_k(A) = \{w : \exists \beta, \gamma \text{ t.q. } S \rightarrow \beta A \gamma \text{ et } w \in \text{Premier}_k(\gamma)\}$

Proposition

Une grammaire est $LL(k)$ si pour toute paire de productions $A \rightarrow \alpha$ et $A \rightarrow \beta$, on a :

$$\text{Premier}_k(\alpha \text{ Suivant}_k(A)) \cap \text{Premier}_k(\beta \text{ Suivant}_k(A)) = \emptyset$$

Analyse $LL(k)$

Exemple

$$\begin{cases} S \rightarrow abX \mid \varepsilon \\ X \rightarrow Saa \mid b \end{cases} \quad \text{n'est pas une grammaire } LL(1)$$

	Effaçable	Premier	Suivant
S	oui	a	$\$ a$
X	non	$a b$	$\$ a$

Conflit Premier/Suivant : $\text{Premier}(abX) \cap \text{Suivant}(S) \neq \emptyset$

	$\$$	a	b
S	$S \rightarrow \varepsilon$	$S \rightarrow abX$ $S \rightarrow \varepsilon$	
X		$X \rightarrow Saa$	$X \rightarrow b$

Analyse $LL(k)$

$$\begin{cases} S \rightarrow abX \mid \varepsilon \\ X \rightarrow Saa \mid b \end{cases} \text{ est une grammaire } LL(2)$$

	Effaçable	Premier ₂	Suivant ₂
S	oui	ab	\$ aa
X	non	$aa \quad ab \quad b$	\$ aa

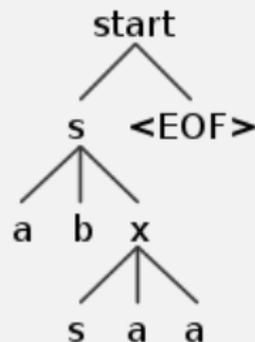
En prévisualisant deux lettres, il n'y a plus de conflit :
 $\text{Premier}_2(abX) \cap \text{Suivant}_2(S) = \emptyset$

	\$	aa	ab	b
S	$S \rightarrow \varepsilon$	$S \rightarrow \varepsilon$	$S \rightarrow abX$	
X		$X \rightarrow Saa$	$X \rightarrow Saa$	$X \rightarrow b$

Analyse $LL(k)$

	\$	aa	ab	b
S	$S \rightarrow \epsilon$	$S \rightarrow \epsilon$	$S \rightarrow abX$	
X		$X \rightarrow Saa$	$X \rightarrow Saa$	$X \rightarrow b$

Mot analysé	Pile	
abaa\$	S\$	$S \rightarrow abX$
abaa\$	abX\$	assortiment
abaa\$	bX\$	assortiment
abaa\$	X\$	$X \rightarrow Saa$
abaa\$	Saa\$	$S \rightarrow \epsilon$
abaa\$	aa\$	assortiment
abaa\$	a\$	assortiment
abaa\$	\$	SUCCES



Fait

Il existe des grammaires ni ambiguës, ni récursives gauche qui ne sont $LL(k)$ pour aucun k .

Exemple

$$\left\{ \begin{array}{l} S \rightarrow A \mid B \\ A \rightarrow aAb \mid c \\ B \rightarrow aBbb \mid d \end{array} \right.$$

$a^k \in \text{Premier}_k(A) \cap \text{Premier}_k(B)$ pour tout k

\leadsto Conflit Premier/Premier pour tout k

Exemple

$$\{ S \rightarrow aSb \mid bSa \mid \varepsilon \}$$

$\text{Premier}_k(aSb) \cap \text{Suivant}_k(S) = \{a\}\{a, b\}^{k-2}\{b\}$ pour tout k

\leadsto Conflit Premier/Suivant pour tout k

Analyse syntaxique avec ANTLR

ANTLR4 met en œuvre une analyse descendante prédictive qui a les caractéristiques suivantes

- factorisation à gauche de la grammaire automatique
- suppression automatique des récursivités gauches immédiates (mais pas des récursivités gauches indirectes, e.g. $A \rightarrow B\alpha$, $B \rightarrow A\beta$)
- résolution de certaines ambiguïtés
 - en jouant sur l'ordre des productions pour lever celles liées aux priorités des opérateurs
 - pour les ambiguïtés dues à l'associativité, par défaut l'association s'effectue à gauche ou de façon explicite à droite, e.g. l'exponentielle, $expr : expr \wedge \langle assoc = right \rangle expr$
- basé sur une stratégie $LL(k)$
- enrichi d'un mécanisme additionnel qui permet de traiter plus que les grammaires $LL(k)$ mais qui induit alors un surcoût en temps