Langages et Compilation

Analyse descendante prédictive

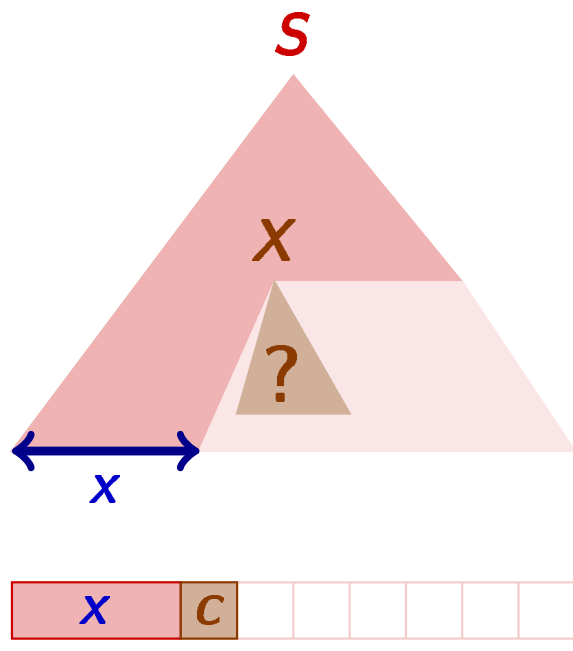
Introduction

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

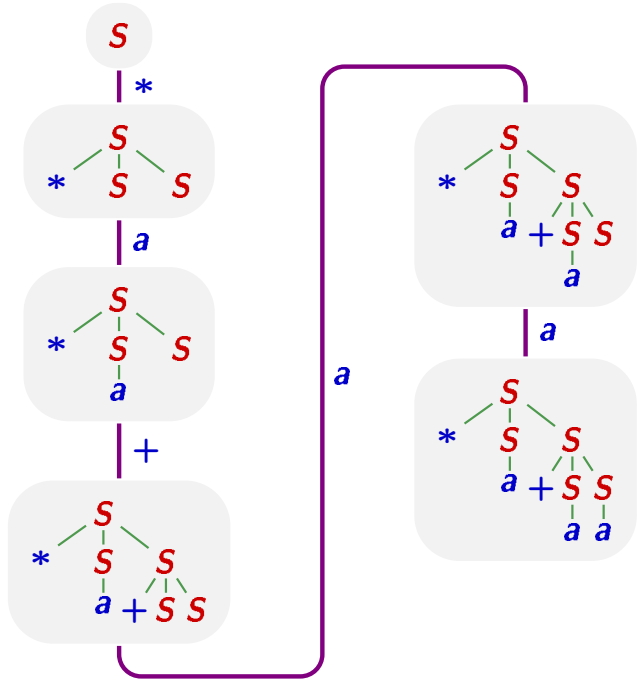
Exemple $S \to +SS \mid *SS \mid a$

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	$S \to +SS$	$S \to *SS$	$S \to a$

Analyse du mot $*a+aa$



Introduction

Exemple $S \to aSbS \mid \varepsilon$

Le choix devient plus délicat lorsque la grammaire comprend des ε-productions. Quel critère doit-on prendre en compte pour choisir entre la règle $S \to aSbS$ et la règle $S \to \varepsilon$?
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.
$\rightsquigarrow$ On sait alors déterminer la règle à appliquer au vu du terminal courant.

	\$	***a***	***b***
S	$S \to \varepsilon$	$S \to aSbS$	$S \to \varepsilon$

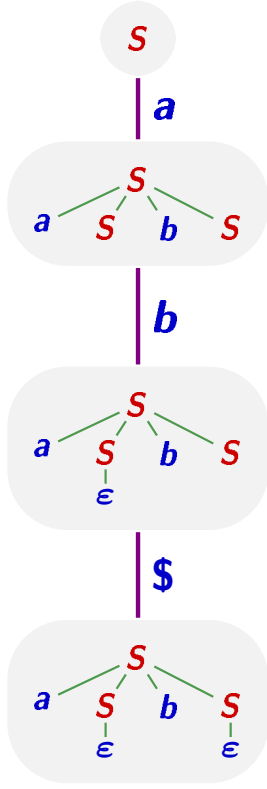
Introduction

Exemple $S \to aSbS \mid \varepsilon$

sa table d'analyse

	$	a	b
S	$S \to \varepsilon$	$S \to aSbS$	$S \to \varepsilon$

Analyse du mot $ab\$$



Introduction

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 (**L**eft to right scanning)*
- **L** *construction d'une dérivation gauche (**L**eft 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

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 $\mathbf{Eff}_i$ des variables A.

$$\mathbf{Eff}_0 = \{A \in N : A \to \varepsilon \in P\}$$

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

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

Exemple

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

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

Les ensembles Premier

Soit α une chaîne de $(\Sigma \cup N)^*$. $\mathbf{Premier}(\alpha)$ 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 :

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

Pour une variable X :

$$\mathbf{Premier}(X) = \bigcup_{X \to \beta \in P} \mathbf{Premier}(\beta)$$

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

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

Les ensembles Premier

Exemple

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

Calcul des ensembles Premier pour les variables :

$$\begin{array}{lcll} \mathbf{Premier(S)} & = & \text{Premier}(XY) & S \to XY \\ & = & \text{Premier}(X) \cup \text{Premier}(Y) & X \text{ effaçable} \\ \mathbf{Premier(X)} & = & \text{Premier}(aXb) & X \to aXb \mid \varepsilon \\ & = & \{a\} & \\ \mathbf{Premier(Y)} & = & \text{Premier}(cZ) \cup \text{Premier}(Ze) & Y \to cZ \mid Ze \\ & = & \{c\} \cup \text{Premier}(Z) \cup \{e\} & Z \text{ effaçable} \\ \mathbf{Premier(Z)} & = & \text{Premier}(dcZ) & Z \to dcZ \mid \varepsilon \\ & = & \{d\} & \end{array}$$

Les ensembles Suivant

Soit X une variable. $\mathbf{Suivant}(X)$ 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.

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

Calcul des ensembles Suivant

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

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

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

Les ensembles Suivant

Exemple

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

Calcul des ensembles Suivant :

Suivant(S) contient \$

Suivant(X)

$S \to XY \rightsquigarrow \text{Premier}(Y\ \text{Suivant}(S)) \subseteq \text{Suivant}(X)$

$\rightsquigarrow \text{Premier}(Y) \subseteq \text{Suivant}(X)$ (Y non effaçable)

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

Suivant(Y)

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

Suivant(Z)

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

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

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

Les variables Effaçables et les ensembles Premier et Suivant

Exemple

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

Bilan des calculs :

	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

Construction de la table d'analyse

Table à deux dimensions indexée par les variables et les terminaux

Pour toute production $X \to \alpha$:

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

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

	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

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

Grammaire $LL(1)$

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 $A \to \alpha$ et $A \to \beta$, on a :

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

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

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

Grammaire $LL(1)$

Exemple

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

est une grammaire $LL(1)$

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

$\rightsquigarrow$ Le choix de la règle à appliquer se fait de façon déterministe.

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

Grammaire $LL(1)$

Exemple

$$\begin{cases} S & \to & XY \mid Z \\ X & \to & aXb \mid \varepsilon \\ Y & \to & bX \\ Z & \to & bZc \mid c \end{cases}$$

n'est pas une grammaire $LL(1)$

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

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

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

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

	\$	b	$\cdots$
$\cdots$			
S		$S \to XY$ $S \to Z$	

Grammaire $LL(1)$

Exemple

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

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

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

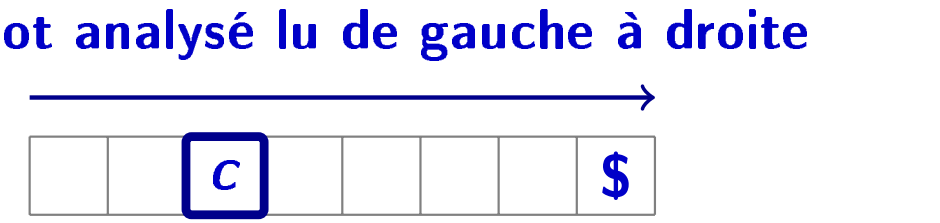
$\text{Suivant}(X) = \{b\}$

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

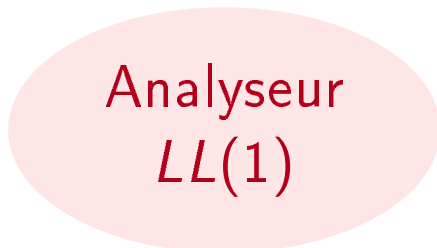
	\$	b	$\cdots$
$\cdots$			
X		$X \to bX$ $X \to \varepsilon$	

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.



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



La pile

S

L'arbre d'analyse
à construire

Analyseur $LL(1)$

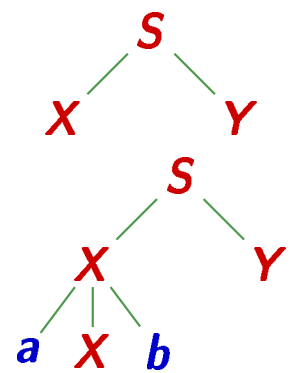
À chaque étape, on examine le terminal courant c du mot analysé et le sommet de la pile (premier symbole non traité de l'arbre en construction).

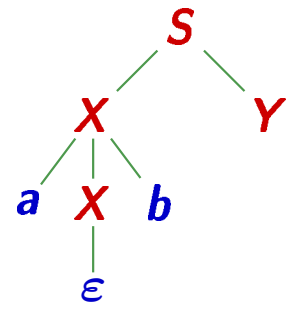
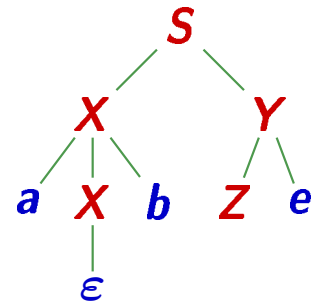
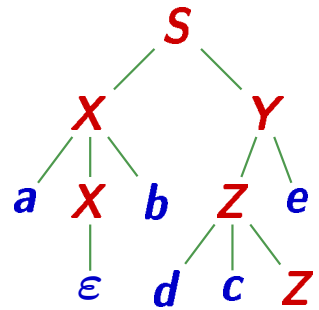
- Soit le sommet de la pile est un terminal a :
  - si $a \neq c$,
    l'analyse s'arrête et retourne ÉCHEC
  - si $a = c = \$$,
    l'analyse s'arrête et retourne SUCCÈS
  - si $a = c \neq \$$,
    a est dépilé et on avance dans la lecture du mot analysé
- Soit le sommet de la pile est une variable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

Analyseur $LL(1)$

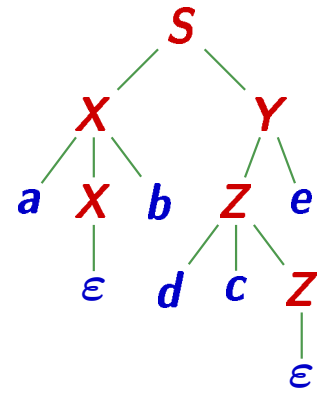
	$	a	b	c	d	e
S		$S \to XY$		$S \to XY$	$S \to XY$	$S \to XY$
X		$X \to aXb$	$X \to \varepsilon$	$X \to \varepsilon$	$X \to \varepsilon$	$X \to \varepsilon$
Y				$Y \to cZ$	$Y \to Ze$	$Y \to Ze$
Z	$Z \to \varepsilon$				$Z \to dcZ$	$Z \to \varepsilon$

Mot analysé	Pile	
$abdce\$$	$S\$$	$S \to XY$
$abdce\$$	$XY\$$	$X \to aXb$
$abdce\$$	$aXbY\$$	assortiment



Mot analysé	Pile		
$a\mathbf{b}dce\,\$$	$\mathbf{X}\,bY\,\$$	$X \to \varepsilon$	
$a\mathbf{b}dce\,\$$	$\mathbf{b}Y\,\$$	assortiment	
$ab\mathbf{d}ce\,\$$	$\mathbf{Y}\,\$$	$Y \to Ze$	
$ab\mathbf{d}ce\,\$$	$\mathbf{Z}e\,\$$	$Z \to dcZ$	
$ab\mathbf{d}ce\,\$$	$\mathbf{d}cZe\,\$$	assortiment	

Mot analysé	Pile	
$abdce\$$	$cZe\$$	assortiment
$abdce\$$	$Ze\$$	$Z \to \varepsilon$
$abdce\$$	$e\$$	assortiment
$abdce\$$	$\$$	SUCCES

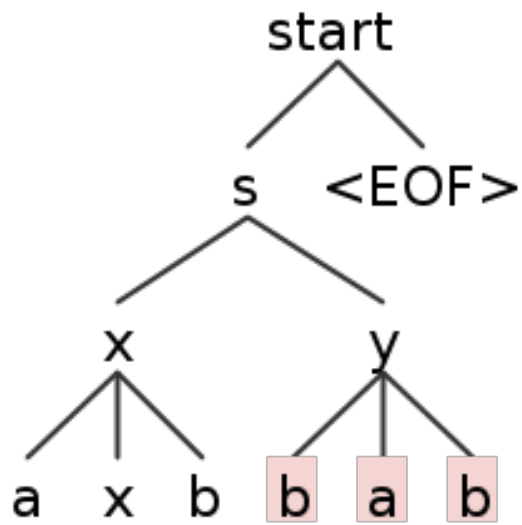


Le coût de l'analyse du mot est linéaire en la taille de l'arbre.

Analyseur $LL(1)$

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

Mot analysé	Pile	
$abbab\$$	$S\$$	$S \to XY$
$abbab\$$	$XY\$$	$X \to aXb$
$abbab\$$	$aXbY\$$	assortiment
$abbab\$$	$XbY\$$	$X \to \varepsilon$
$abbab\$$	$bY\$$	assortiment
$abbab\$$	$Y\$$	ECHEC



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

$$\begin{cases} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & (E) \mid nb \end{cases}$$

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 $LL(1)$

$$\begin{cases} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & (E) \mid nb \end{cases} \longrightarrow \begin{cases} E & \rightarrow & TY \\ Y & \rightarrow & +TY \mid \varepsilon \\ T & \rightarrow & FZ \\ Z & \rightarrow & *FZ \mid \varepsilon \\ F & \rightarrow & (E) \mid nb \end{cases}$$

Rendre une grammaire $LL(1)$

Par élimination de la récursivité gauche

$$\begin{cases} E & \to & TY \\ Y & \to & +TY \mid \varepsilon \\ T & \to & FZ \\ Z & \to & *FZ \mid \varepsilon \\ F & \to & (E) \mid nb \end{cases}$$

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

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

Rendre une grammaire $LL(1)$

Par substitution et factorisation

Exemple

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

n'est pas $LL(1)$

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

	\$	id	$\cdots$
$\cdots$			
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.

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

- Substitution

$$
\begin{array}{lcl}
E & \rightarrow & TR \\
T & \rightarrow & id \mid (E) \mid id[E] \\
R & \rightarrow & +E \mid \varepsilon
\end{array}
$$

- Factorisation à gauche

$$
\begin{array}{lcl}
E & \rightarrow & TR \\
T & \rightarrow & id\ X \mid (E) \\
X & \rightarrow & [E] \mid \varepsilon \\
R & \rightarrow & +E \mid \varepsilon
\end{array}
$$

Rendre une grammaire $LL(1)$

Par substitution et factorisation

On obtient une grammaire $LL(1)$

$$\begin{cases} E & \to & TR \\ T & \to & id\,X \mid (E) \\ X & \to & [E] \mid \varepsilon \\ R & \to & +E \mid \varepsilon \end{cases}$$

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

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

Rendre une grammaire $LL(1)$

ou rendre l'analyse déterministe

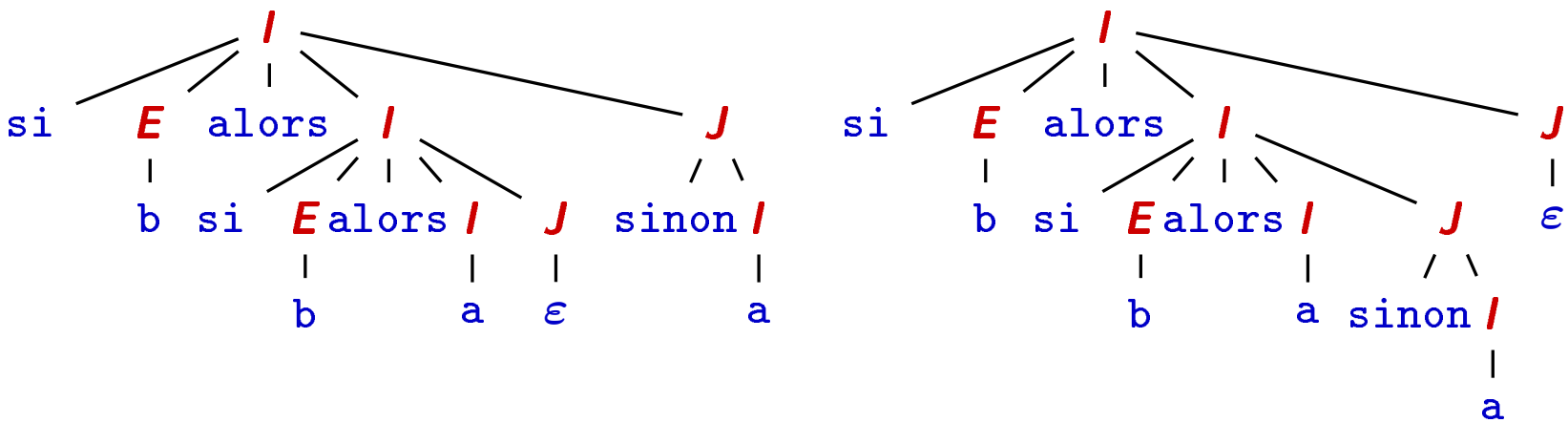
La grammaire des instructions de branchements conditionnels

$$\begin{cases} I & \to & si\ E\ alors\ I\ sinon\ I \mid si\ E\ alors\ I \mid a \\ E & \to & b \end{cases}$$

même factorisée

$$\begin{cases} I & \to & si\ E\ alors\ I\ J \mid a \\ J & \to & sinon\ I \mid \varepsilon \\ E & \to & b \end{cases}$$

n'est pas $LL(1)$ car ambiguë.



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

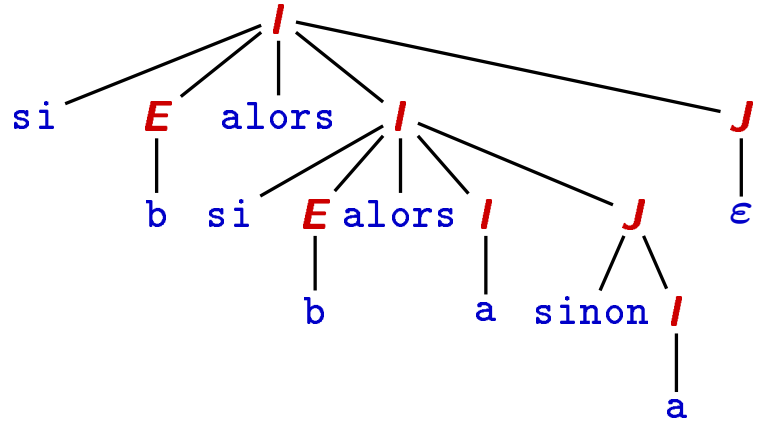
Rendre une grammaire $LL(1)$

ou rendre l'analyse déterministe

L'ambiguïté engendre un conflit Premier/Suivant pour les règles $J \to \textit{sinon}\ I$ et $J \to \varepsilon$

	$	*sinon*	···
···			
J	$J \to \varepsilon$	$J \to \textit{sinon}\ I$ $J \to \varepsilon$	···

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 $J \to \textit{sinon}\ I$ au détriment de la règle $J \to \varepsilon$.



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 :

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

Proposition

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

$$\mathbf{Premier_k}\big(\alpha\, \mathbf{Suivant_k(A)}\big) \cap \mathbf{Premier_k}\big(\beta\, \mathbf{Suivant_k(A)}\big) = \emptyset$$

Analyse $LL(k)$

Exemple

$$\begin{cases} S & \to & abX \mid \varepsilon \\ X & \to & Saa \mid b \end{cases}$$

n'est pas une grammaire $LL(1)$

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

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

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

Analyse $LL(k)$

$$\begin{cases} S & \to & abX \mid \varepsilon \\ X & \to & Saa \mid b \end{cases}$$

est une grammaire $LL(2)$

	Effaçable	Premier_2	Suivant_2
S	oui	ab	\$ aa
X	non	aa ab b	\$ aa

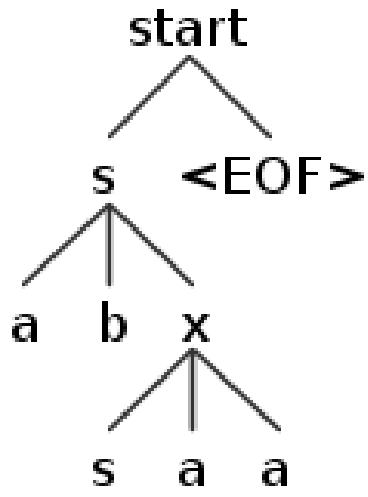
En prévisualisant deux lettres, il n'y a plus de conflit :
$\text{Premier}_2(abX) \cap \text{Suivant}_2(S) = \emptyset$

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

Analyse $LL(k)$

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

Mot analysé	Pile	
$abaa\,\$$	$S\,\$$	$S \to abX$
$abaa\,\$$	$abX\,\$$	assortiment
$abaa\,\$$	$bX\,\$$	assortiment
$abaa\,\$$	$X\,\$$	$X \to Saa$
$abaa\,\$$	$Saa\,\$$	$S \to \varepsilon$
$abaa\,\$$	$aa\,\$$	assortiment
$abaa\,\$$	$a\,\$$	assortiment
$abaa\,\$$	$\$$	SUCCES



Analyse $LL(k)$

Fait

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

Exemple

$$\begin{cases} S & \to & A \mid B \\ A & \to & aAb \mid c \\ B & \to & aBbb \mid d \end{cases}$$

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

$\rightsquigarrow$ Conflit Premier/Premier pour tout k

Exemple

$$\{ \quad S \quad \to \quad aSb \mid bSa \mid \varepsilon$$

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

$\rightsquigarrow$ 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. $A \to B\alpha$, $B \to A\beta$)
- 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