Langages et Compilation

Analyse syntaxique ascendante

Analyse syntaxique ascendante

Construction de l'arbre de dérivation selon l'**ordre postfixé** :
on part des feuilles (les terminaux) et on remonte à la racine.

Deux opérations :

- la création des feuilles par **décalage**
- la construction des nœuds internes par **réduction**.

Analyse syntaxique ascendante

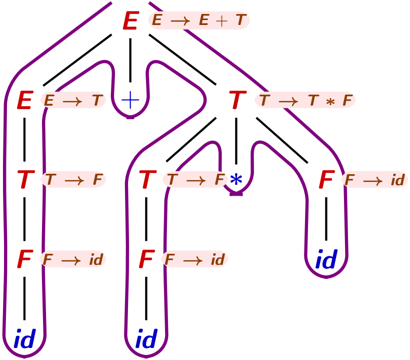
$$\begin{cases} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & id \end{cases}$$

Les opérations de décalage et de réduction s'effectuent dans l'ordre du parcours postfixé de l'arbre de dérivation.

id, $F \rightarrow id$, $T \rightarrow F$, $E \rightarrow T$, $+$, id, $F \rightarrow id$, $T \rightarrow F$, $*$, id, $F \rightarrow id$, $T \rightarrow T * F$, $E \rightarrow E + T$

La suite inverse des réductions dans l'ordre du parcours postfixé donne la dérivation droite du mot analysé.

Analyse du mot $id + id * id$



$E \rightarrow E + T \rightarrow E + T * F \rightarrow E + T * id \rightarrow E + F * id \rightarrow E + id * id \rightarrow T + id * id \rightarrow F + id * id \rightarrow id + id * id$

Analyse syntaxique ascendante

Approche non-déterministe

En pratique on utilise une pile pour réaliser une analyse ascendante par décalage/réduction.
Cette pile contient au départ un marqueur particulier $\$$.

L'analyseur lit le mot à analyser (complété du marqueur $\$$) de gauche à droite.

Il opère

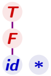
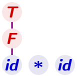
- soit en décalant un caractère du mot vers la pile
- soit par réduction, à condition qu'il trouve au sommet de la pile une chaîne β correspondant à la partie droite d'une production $\boldsymbol{A} \rightarrow \beta$ qu'on remplace alors par $\boldsymbol{A}$

et ceci de façon non déterministe en devinant les bonnes opérations.

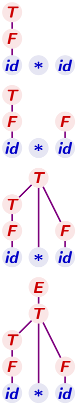
Analyse syntaxique ascendante

Approche non-déterministe

Analyse du mot $\boldsymbol{id * id}$ en devinant les bonnes opérations.

Mot	Pile		Arbre
$id * id\$$	$\$$	decaler	
$id * id\$$	$\$id$	reduire $F \rightarrow id$	
$id * id\$$	$\$F$	reduire $T \rightarrow F$	
$id * id\$$	$\$T$	decaler	
$id * id\$$	$\$T *$	decaler	

$\vdots$	$\vdots$	
$id * id\$$	$\$T *$	decaler
$id * id\$$	$\$T * id$	reduire $F \rightarrow id$
$id * id\$$	$\$T * F$	reduire $T \rightarrow T * F$
$id * id\$$	$\$T$	reduire $E \rightarrow T$
$id * id\$$	$\$E$	fin



Automate caractéristique

L'analyseur doit identifier au sommet de la pile les chaînes qui sont partie droite d'une production de la grammaire et ceci de manière efficace.

À cet effet, on construit un automate fini qui va décrire le comportement au sommet de la pile.

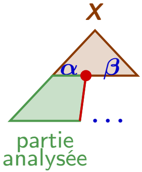
Une convention utile par la suite est d'augmenter la grammaire d'une nouvelle variable **init** et de la règle **init** $\rightarrow$ $\boldsymbol{S}\bullet$.
init est alors le nouvel axiome et remplace l'ancien axiome $\boldsymbol{S}$.

Les éléments de l'automate

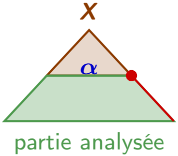
Item $LR(0)$

$X \to \alpha \bullet \beta$ où $X \to \alpha\beta$ est une règle de la grammaire et $\bullet$ est un symbole particulier

ce qui a déjà été analysé $\bullet$ *ce qu'on attend*



Les items $X \to \alpha\bullet$ sont dits **complets**



On a quatre items $LR(0)$ associés à la production $A \to Abc$:
$A \to \bullet Abc$, $A \to A \bullet bc$, $A \to Ab \bullet c$ et l'item complet $A \to Abc\bullet$
Le seul item associé à une ε-production $A \to \varepsilon$ est l'item complet $A \to \bullet$

L'automate fini caractéristique

L'AFN avec ε-transitions qui caractérise les items valides :

l'alphabet : $\Sigma \cup N$

les états : les items $LR(0)$

les états d'acceptation : les items $LR(0)$ complets

l'état initial : l'item init $\to \bullet S$

la fonction de transition : trois types de transition

- décalage $\quad \boxed{X \to \alpha \bullet a\beta} \xrightarrow{a} \boxed{X \to \alpha a \bullet \beta}$

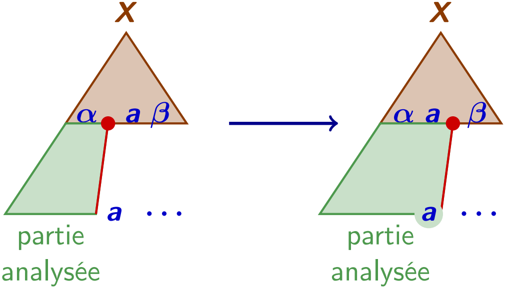
- réduction $\quad \boxed{X \to \alpha \bullet Y\beta} \xrightarrow{Y} \boxed{X \to \alpha Y \bullet \beta}$

- prédiction $\quad \boxed{X \to \alpha \bullet Y\beta} \xrightarrow{\varepsilon} \boxed{Y \to \bullet\delta}$

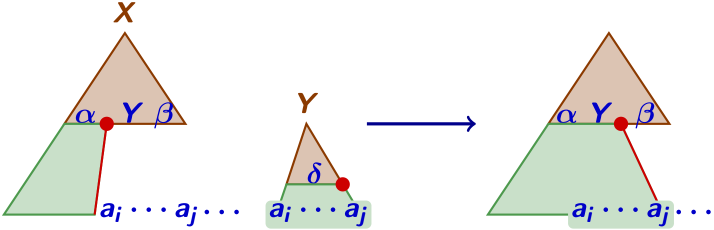
pour toute production $Y \to \delta$

Les transitions

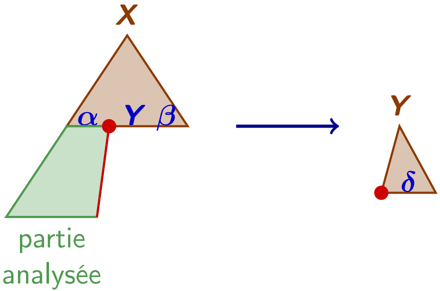
- **décalage**



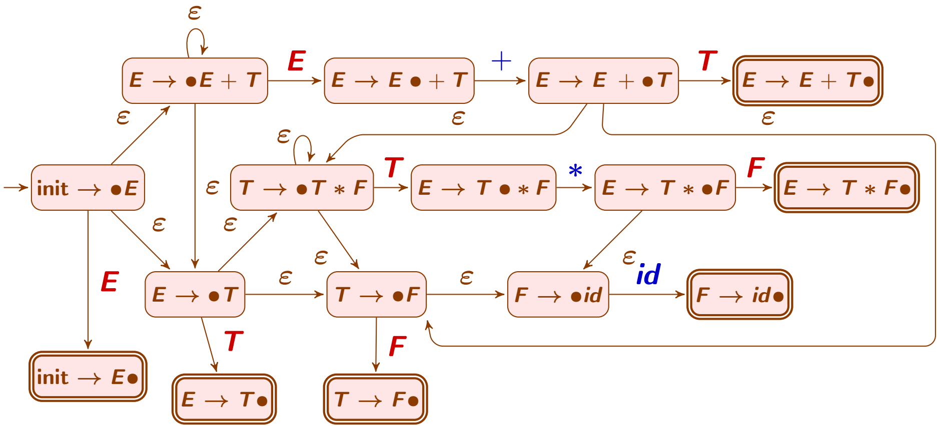
- **réduction**



- **prédiction**



l'AFN avec ε-transitions



La grammaire augmentée
$$\begin{cases} \text{init} & \to & E \\ E & \to & E + T \mid T \\ T & \to & T * F \mid F \\ F & \to & id \end{cases}$$

Déterminisation de l'AFN avec ε-transitions

Les états de l'AFD sont des ensembles d'items $LR(0)$.

On utilise l'opération de clôture pour supprimer les ε-transitions : `Cloture`$(\mathcal{I})$ l'ensemble des items accessibles à partir d'un item de $\mathcal{I}$ par des ε-transitions

> *c'est le plus petit ensemble contenant $\mathcal{I}$ et tel que si $X \to \alpha \bullet Y\beta$ est dans $\mathcal{I}$ et $Y \to \gamma$ est une production de la grammaire alors $Y \to \bullet\gamma$ est dans `Cloture`$(\mathcal{I})$*

La **transition** étiquetée par un terminal ou une variable est définie ainsi

$$\delta(\mathcal{I}, e) = \mathtt{Cloture}\big(\{X \to \alpha e \bullet \beta : X \to \alpha \bullet e\beta \in \mathcal{I}\}\big)$$

L'état initial est $\mathtt{Cloture}\big(\{\mathbf{init} \to \bullet S\}\big)$.

Seuls les états non vides accessibles à partir de l'état initial via la fonction δ sont construits.

Les états d'acceptation sont ceux qui contiennent un item complet.

Les états de l'AFD

0 **état initial**

$$\boxed{\begin{array}{c} \text{init} \rightarrow \bullet E \\ \hline E \rightarrow \bullet E + T \\ E \rightarrow \bullet T \\ T \rightarrow \bullet T * F \\ T \rightarrow \bullet F \\ F \rightarrow \bullet id \end{array}} \left.\begin{array}{c} \\ \\ \\ \\ \end{array}\right\} \begin{array}{l} \text{items} \\ \text{hérités} \\ \text{par} \\ \text{clôture} \end{array}$$

$1 = \delta(0, E)$

$$\boxed{\begin{array}{c} \text{init} \rightarrow E\bullet \\ E \rightarrow E \bullet + T \end{array}}$$

$2 = \delta(0, T)$

$$\boxed{\begin{array}{c} E \rightarrow T\bullet \\ T \rightarrow T \bullet * F \end{array}}$$

$3 = \delta(0, F)$

$$\boxed{T \rightarrow F\bullet}$$

$4 = \delta(0, id)$

$$\boxed{F \rightarrow id\bullet}$$

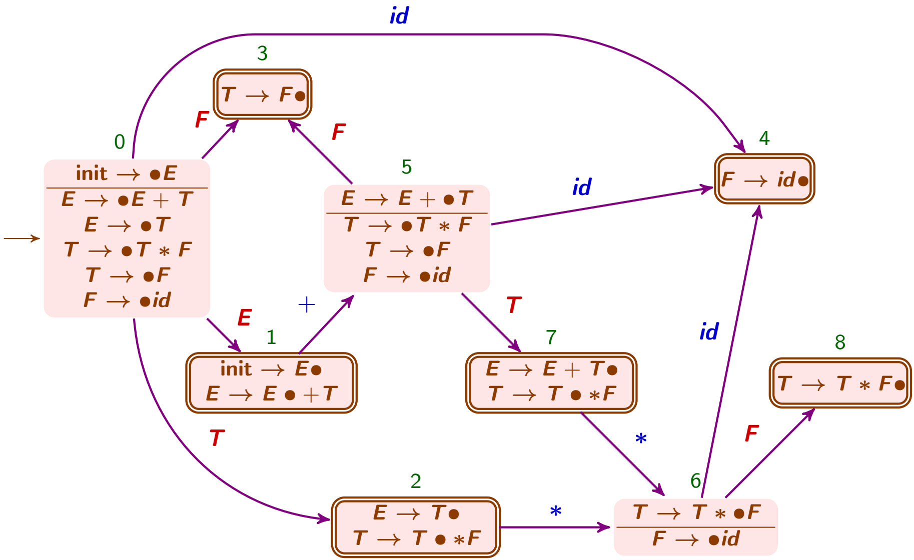
$$5 = \delta(1, +) \qquad \boxed{\begin{array}{c} E \to E + \bullet T \\ \hline T \to \bullet T * F \\ T \to \bullet F \\ F \to \bullet id \end{array}} \; \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} \text{clôture}$$

$$6 = \delta(2, *) \qquad \boxed{\begin{array}{c} T \to T * \bullet F \\ \hline F \to \bullet id \end{array}} \; \left.\begin{array}{c} \\ \end{array}\right\} \text{clôture}$$

$$7 = \delta(5, \boldsymbol{T}) \qquad \boxed{\boxed{\begin{array}{c} E \to E + T \bullet \\ T \to T \bullet * F \end{array}}} \qquad \delta(5, \boldsymbol{F}) = 3 \text{ , } \delta(5, \boldsymbol{id}) = 4$$

$$8 = \delta(6, \boldsymbol{F}) \qquad \boxed{\boxed{T \to T * F \bullet}} \qquad \delta(6, \boldsymbol{id}) = 4$$

L'AFD



id
3
T → F●
F
F
0
init → ●E
E → ●E + T
E → ●T
T → ●T ∗ F
T → ●F
F → ●id
5
E → E + ●T
T → ●T ∗ F
T → ●F
F → ●id
id
4
F → id●
E
+
1
init → E●
E → E ● +T
T
7
E → E + T●
T → T ● ∗F
id
8
T → T ∗ F●
T
∗
F
2
E → T●
T → T ● ∗F
∗
6
T → T ∗ ●F
F → ●id

Construction de la table d'analyse $\boldsymbol{LR}(0)$

La table a en ligne les états de l'automate et en colonne le marqueur $\$$, les terminaux et variables de la grammaire.

Pour chaque état $\mathcal{I}$ de l'automate :

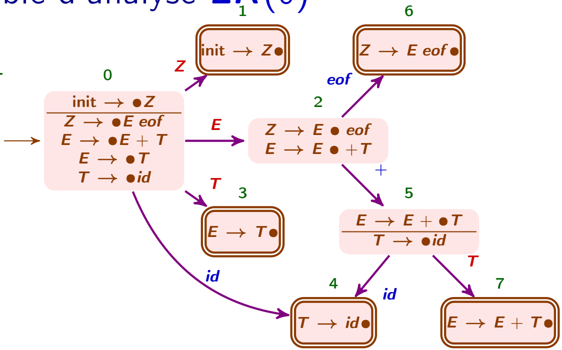
- pour chaque transition $\mathcal{I} \xrightarrow{a} \mathcal{J}$ étiquetée par un terminal a ajouter l'action **décaler** $\mathcal{J}$ à l'entrée table$[\mathcal{I}, a]$
- pour chaque transition $\mathcal{I} \xrightarrow{X} \mathcal{J}$ étiquetée par une variable X, enregistrer $\mathcal{J}$ à l'entrée table$[\mathcal{I}, X]$

Pour chaque état d'acceptation $\mathcal{I}$ de l'automate :

- Si $\mathcal{I}$ contient l'item complet **init** $\rightarrow S\bullet$ ajouter l'action **accepter** à l'entrée table$[\mathcal{I}, \$]$ et l'action **rejeter** à l'entrée table$[\mathcal{I}, \$]$ pour tout terminal a
- Pour tout autre item complet $X \rightarrow \alpha\bullet$ de $\mathcal{I}$, ajouter l'action **réduire** $X \rightarrow \alpha$ aux entrées table$[\mathcal{I}, a]$ pour tout a terminal ou marqueur de fin de mot

Construction de la table d'analyse $LR(0)$

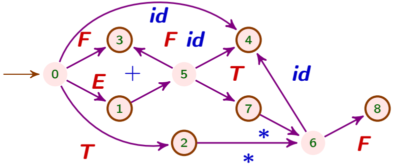
$$\begin{cases} Z & \rightarrow & E\ eof \\ E & \rightarrow & E + T \mid T \\ T & \rightarrow & id \end{cases}$$



	\$	*eof*	+	*id*	*E*	*T*	*F*
0				**d** 4	2	3	1
1	accepter	rejeter	rejeter	rejeter			
2		**d** 6	**d** 5				
3	r $E \rightarrow T$	r $E \rightarrow T$	r $E \rightarrow T$	r $E \rightarrow T$			
4	r $T \rightarrow id$	r $T \rightarrow id$	r $T \rightarrow id$	r $T \rightarrow id$			
5				**d** 4		7	
6	r $Z \rightarrow E\ eof$	r $Z \rightarrow E\ eof$	r $Z \rightarrow E\ eof$	r $Z \rightarrow E\ eof$			
7	r $E \rightarrow E + T$	r $E \rightarrow E + T$	r $E \rightarrow E + T$	r $E \rightarrow E + T$			

La table d'analyse $\boldsymbol{LR}(0)$

$$\begin{cases} \text{init} & \to & E \\ E & \to & E + T \mid T \\ T & \to & T * F \mid F \\ F & \to & id \end{cases}$$



	\$	$+$	$*$	$\boldsymbol{id}$	$\boldsymbol{E}$	$\boldsymbol{T}$	$\boldsymbol{F}$
0				**d** 4	1	2	3
1	accepter	rejeter **d** 5	rejeter	rejeter			
2	**r** $\boldsymbol{E \to T}$	**r** $\boldsymbol{E \to T}$	**r** $\boldsymbol{E \to T}$ **d** 6	**r** $\boldsymbol{E \to T}$			
3	**r** $\boldsymbol{T \to F}$	**r** $\boldsymbol{T \to F}$	**r** $\boldsymbol{T \to F}$	**r** $\boldsymbol{T \to F}$			
4	**r** $\boldsymbol{F \to id}$	**r** $\boldsymbol{F \to id}$	**r** $\boldsymbol{F \to id}$	**r** $\boldsymbol{F \to id}$			
5				**d** 4		7	3
6				**d** 4			8
7	**r** $\boldsymbol{E \to E + T}$	**r** $\boldsymbol{E \to E + T}$	**r** $\boldsymbol{E \to E + T}$ **d** 6	**r** $\boldsymbol{E \to E + T}$			
8	**r** $\boldsymbol{T \to T * F}$	**r** $\boldsymbol{T \to T * F}$	**r** $\boldsymbol{T \to T * F}$	**r** $\boldsymbol{T \to T * F}$			

Grammaire $LR(0)$

Définition

Une grammaire est $LR(0)$ s'il existe au plus une action par entrée dans la table $LR(0)$.

Une grammaire ne sera pas $LR(0)$ si l'on a :

- soit un **conflit décaler/réduire**
  un état de l'AFD caractéristique contient conjointement
  un item non complet $X \to \alpha \bullet a\beta$ où a est un terminal
  et un item complet $Y \to \gamma\bullet$
- soit un **conflit réduire/réduire**
  un état de l'AFD caractéristique contient deux items complets
  $X \to \alpha\bullet$ et $Y \to \beta\bullet$

pas de conflit décaler/décaler possible, l'automate est déterministe

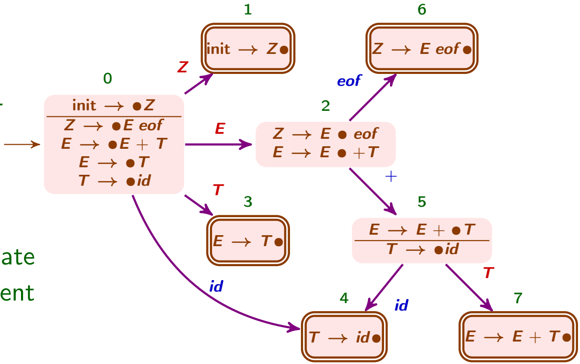
Grammaire *LR*(0)

La grammaire

$$\begin{cases} Z & \rightarrow & E\ eof \\ E & \rightarrow & E + T \mid T \\ T & \rightarrow & id \end{cases}$$

est $LR(0)$.

Les états de son automate caractéristique n'induisent pas de conflit.



	$	*eof*	+	*id*	*E*	*T*	*F*
0				d 4	2	3	1
1	accepter	rejeter	rejeter	rejeter			
2		d 6	d 5				
3	r $E \rightarrow T$	r $E \rightarrow T$	r $E \rightarrow T$	r $E \rightarrow T$			
4	r $T \rightarrow id$	r $T \rightarrow id$	r $T \rightarrow id$	r $T \rightarrow id$			
5				d 4		7	
6	r $Z \rightarrow E\ eof$	r $Z \rightarrow E\ eof$	r $Z \rightarrow E\ eof$	r $Z \rightarrow E\ eof$			
7	r $E \rightarrow E + T$	r $E \rightarrow E + T$	r $E \rightarrow E + T$	r $E \rightarrow E + T$			

L'analyse est alors déterministe car une action au plus est possible à chaque étape.

Grammaire $LR(0)$

La grammaire $\begin{cases} E & \rightarrow & E+T \mid T \\ T & \rightarrow & T*F \mid F \\ F & \rightarrow & id \end{cases}$ n'est pas $LR(0)$

Trois états engendrent des conflits du type décaler/réduire.

$\boxed{\begin{array}{c} \text{init} \rightarrow E\bullet \\ E \rightarrow E\bullet + T \end{array}}$ $\boxed{\begin{array}{c} E \rightarrow T\bullet \\ T \rightarrow T\bullet * F \end{array}}$ $\boxed{\begin{array}{c} E \rightarrow E+T\bullet \\ T \rightarrow T\bullet * F \end{array}}$

	\$	$+$	$*$	id	E	T	F
0				d 4	1	2	3
1	accepter	rejeter d 5	rejeter	rejeter			
2	r $E \rightarrow T$	r $E \rightarrow T$	r $E \rightarrow T$ d 6	r $E \rightarrow T$			
3	r $T \rightarrow F$	r $T \rightarrow F$	r $T \rightarrow F$	r $T \rightarrow F$			
4	r $F \rightarrow id$	r $F \rightarrow id$	r $F \rightarrow id$	r $F \rightarrow id$			
5				d 4		7	3
6				d 4			8
7	r $E \rightarrow E+T$	r $E \rightarrow E+T$	r $E \rightarrow E+T$ d 6	r $E \rightarrow E+T$			
8	r $T \rightarrow T*F$	r $T \rightarrow T*F$	r $T \rightarrow T*F$	r $T \rightarrow T*F$			

Analyseur $LR(0)$

Pour examiner un mot, l'analyseur $LR(0)$ utilise la table d'analyse $LR(0)$ et une pile.

Initialement la pile contient l'état initial 0.
Le mot complété du marqueur de fin $ est lu de gauche à droite.

À chaque étape on examine le symbole courant c du mot analysé et l'état q au sommet de la pile

- Si table$[q, c] =$ **décaler** r alors
  empiler c puis r, et avancer dans la lecture du mot analysé
- Si table$[q, c] =$ **réduire** $X \to \alpha$ alors
  dépiler $2|\alpha|$ symboles, étant donné r le nouvel état au sommet de la pile empiler X puis $\delta(r, X)$, et afficher $X \to \alpha$
- Si table$[q, c] =$ **accepter** alors retourner SUCCÈS
- Sinon retourner ÉCHEC

Analyseur $LR(0)$

	\$	*eof*	$+$	*id*	*E*	*T*	*Z*
0				d 4	2	3	1
1	accepter	rejeter	rejeter	rejeter			
2		d 6	d 5				
3	r $E \to T$	r $E \to T$	r $E \to T$	r $E \to T$			
4	r $T \to id$	r $T \to id$	r $T \to id$	r $T \to id$			
5				d 4		7	
6	r $Z \to E\ eof$	r $Z \to E\ eof$	r $Z \to E\ eof$	r $Z \to E\ eof$			
7	r $E \to E + T$	r $E \to E + T$	r $E \to E + T$	r $E \to E + T$			

Mot analysé	Pile	Action
$id + id\ eof\ \$$	0	decaler 4
$id\ + id\ eof\ \$$	0 *id* 4	reduire $T \to id$ $\quad \delta(0, T) = 3$
$id\ + id\ eof\ \$$	0 *T* 3	reduire $E \to T$ $\quad \delta(0, E) = 2$
$id\ + id\ eof\ \$$	0 *E* 2	decaler 5
$id +\ id\ eof\ \$$	0 *E* 2 + 5	decaler 4
$id + id\ \ eof\ \$$	0 *E* 2 + 5 *id* 4	reduire $T \to id$ $\quad \delta(5, T) = 7$
$id + id\ \ eof\ \$$	0 *E* 2 + 5 *T* 7	reduire $E \to E + T$ $\quad \delta(0, T) = 2$
$id + id\ \ eof\ \$$	0 *E* 2	decaler 6
$id + id\ eof\ \ \$$	0 *E* 2 *eof* 6	reduire $Z \to E\ eof$ $\quad \delta(0, Z) = 1$
$id + id\ eof\ \ \$$	0 *Z* 1	accepter

Grammaire ***SLR***

On parle d'analyse *SLR* lorsque les ensembles Suivant sont pris en compte pour déterminer si une action réduire est possible ou non.

Une grammaire est alors dite *SLR* lorsque tous les conflits peuvent être tranchés par l'examen des ensembles Suivant.

$$\begin{cases} E & \rightarrow & E + T \mid T \\ T & \rightarrow & T * F \mid F \\ F & \rightarrow & id \end{cases}$$ n'est pas $LR(0)$

car trois états engendrent des conflits du type décaler/réduire.

$\boxed{\begin{array}{c} \text{init} \rightarrow E\bullet \\ E \rightarrow E \bullet + T \end{array}}$ $\boxed{\begin{array}{c} E \rightarrow T\bullet \\ T \rightarrow T \bullet * F \end{array}}$ $\boxed{\begin{array}{c} E \rightarrow E + T\bullet \\ T \rightarrow T \bullet * F \end{array}}$

En fait, ces conflits peuvent être tranchés par l'examen des ensembles Suivant.

	Suivant		
init	\$		
E	\$	$+$	
T	\$	$+$	$*$
F	\$	$+$	$*$

L'action réduire par une règle $\boldsymbol{X \rightarrow \alpha}$ n'est envisagée que si le prochain symbole d'entrée est dans **Suivant($\boldsymbol{X}$)**.

Construction de la table d'analyse *SLR*

La table a en ligne les états de l'automate et en colonne le marqueur $\$$, les terminaux et variables de la grammaire.

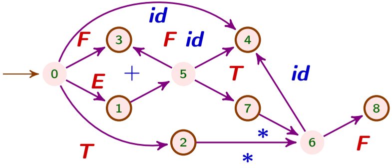
Pour chaque état $\mathcal{I}$ de l'automate :

- pour chaque transition $\mathcal{I} \xrightarrow{a} \mathcal{J}$ étiquetée par un terminal a ajouter l'action **décaler** $\mathcal{J}$ à l'entrée table$[\mathcal{I}, a]$
- pour chaque transition $\mathcal{I} \xrightarrow{X} \mathcal{J}$ étiquetée par une variable X, enregistrer $\mathcal{J}$ à l'entrée table$[\mathcal{I}, X]$

Pour chaque état d'acceptation $\mathcal{I}$ de l'automate :

- Si $\mathcal{I}$ contient l'item complet **init** $\rightarrow S\bullet$ **ajouter l'action accepter à l'entrée table**$[\mathcal{I}, \$]$
- Pour tout autre item complet $X \rightarrow \alpha\bullet$ de $\mathcal{I}$, **ajouter l'action réduire** $X \rightarrow \alpha$ **aux entrées table**$[\mathcal{I}, a]$ **pour tout terminal** a **dans Suivant(**X**)**

La table d'analyse ***SLR***



	Suivant		
init	$		
E	$	$+$	
T	$	$+$	$*$
F	$	$+$	$*$

	$	$+$	$*$	***id***	***E***	***T***	***F***
0				**d** 4	1	2	3
1	accepter	**d** 5					
2	**r** $E \to T$	**r** $E \to T$	**d** 6				
3	**r** $T \to F$	**r** $T \to F$	**r** $T \to F$				
4	**r** $F \to id$	**r** $F \to id$	**r** $F \to id$				
5				**d** 4		7	3
6				**d** 4			8
7	**r** $E \to E + T$	**r** $E \to E + T$	**d** 6				
8	**r** $T \to T * F$	**r** $T \to T * F$	**r** $T \to T * F$				

Analyseur *SLR*

L'analyseur ***SLR*** fonctionne comme l'analyseur ***LR***(0).

	$\$$	$+$	$*$	*id*	*E*	*T*	*F*
0				**d** 4	1	2	3
1	acc	**d** 5					
2	**r** 2	**r** 2	**d** 6				
3	**r** 4	**r** 4	**r** 4				
4	**r** 5	**r** 5	**r** 5				
5				**d** 4		7	3
6				**d** 4			8
7	**r** 1	**r** 1	**d** 6				
8	**r** 3	**r** 3	**r** 4				

1	E	$\rightarrow$	$E + T$
2	E	$\rightarrow$	T
3	T	$\rightarrow$	$T * F$
4	T	$\rightarrow$	F
5	F	$\rightarrow$	id

Mot analysé	Pile	Action
$id + id * id\$$	0	**decaler** 4
$id + id * id\$$	0 id 4	**reduire** $F \rightarrow id$ $\delta(0, F) = 3$
$id + id * id\$$	0 F 3	**reduire** $T \rightarrow F$ $\delta(0, T) = 2$
$id + id * id\$$	0 T 2	**reduire** $E \rightarrow T$ $\delta(0, E) = 1$
$id + id * id\$$	0 E 1	**decaler** 5
$id + id * id\$$	0 E 1 $+$ 5	**decaler** 4
$id + id * id\$$	0 E 1 $+$ 5 id 4	**reduire** $F \rightarrow id$ $\delta(5, F) = 3$

Analyseur *SLR*

$\vdots$	$\vdots$	$\vdots$
$id + id * id\$$	$0\ E\ 1 + 5\ id\ 4$	reduire $F \to id$ $\quad \delta(5, F) = 3$
$id + id * id\$$	$0\ E\ 1 + 5\ F\ 3$	reduire $T \to F$ $\quad \delta(5, T) = 7$
$id + id * id\$$	$0\ E\ 1 + 5\ T\ 7$	decaler 6
$id + id * id\$$	$0\ E\ 1 + 5\ T\ 7 * 6$	decaler 4
$id + id * id\$$	$0\ E\ 1 + 5\ T\ 7 * 6\ id\ 4$	reduire $F \to id$ $\quad \delta(6, F) = 8$
$id + id * id\$$	$0\ E\ 1 + 5\ T\ 7 * 6\ F\ 8$	reduire $T \to T * F$ $\quad \delta(5, T) = 7$
$id + id * id\$$	$0\ E\ 1 + 5\ T\ 7$	reduire $E \to E + T$ $\quad \delta(0, E) = 1$
$id + id * id\$$	$0\ E\ 1$	SUCCES

	\$	+	*	*id*	*E*	*T*	*F*
0				d 4	1	2	3
1	acc	d 5					
2	r 2	r 2	d 6				
3	r 4	r 4	r 4				
4	r 5	r 5	r 5				
5				d 4		7	3
6				d 4			8
7	r 1	r 1	d 6				
8	r 3	r 3	r 4				

1 $E \to E + T$
2 $E \to T$
3 $T \to T * F$
4 $T \to F$
5 $F \to id$

Analyse $\boldsymbol{LR}(1)$

Une méthode qui définit plus finement qu'avec les ensembles Suivant, les lettres pouvant suivre une variable pendant la construction.

On redéfinit l'AFN caractéristique pour l'analyse $LR(1)$

Les états sont maintenant des items $\boldsymbol{LR}(1)$ de la forme

$[\boldsymbol{A} \to \alpha \bullet \beta, \boldsymbol{a}]$ *où* $\boldsymbol{A} \to \alpha\beta$ *une règle de la grammaire et* $\boldsymbol{a}$ *un terminal.*

ce qui a déjà été analysé • ce qu'on attend , a terminal qui peut suivre A

L'état initial est celui qui contient $[\mathbf{init} \to \boldsymbol{S}\bullet, \$]$

Les états d'acceptation sont ceux avec des items $[\boldsymbol{Y} \to \delta\bullet, \boldsymbol{c}]$

Les transitions sont définies ainsi

$$[\boldsymbol{X} \to \alpha \bullet \boldsymbol{a}\beta, \boldsymbol{b}] \xrightarrow{\boldsymbol{a}} [\boldsymbol{X} \to \alpha\boldsymbol{a} \bullet \beta, \boldsymbol{b}]$$

$$[\boldsymbol{X} \to \alpha \bullet \boldsymbol{Y}\beta, \boldsymbol{b}] \xrightarrow{\boldsymbol{Y}} [\boldsymbol{X} \to \alpha\boldsymbol{Y} \bullet \beta, \boldsymbol{b}]$$

$$[\boldsymbol{X} \to \alpha \bullet \boldsymbol{Y}\beta, \boldsymbol{b}] \xrightarrow{\varepsilon} [\boldsymbol{Y} \to \bullet\delta, \boldsymbol{c}]$$

pour toute production $\boldsymbol{Y} \to \delta$ et **tout $\boldsymbol{c}$ dans Premier($\beta\boldsymbol{b}$)**

Analyse $\boldsymbol{LR}(1)$

On déterminise ensuite l'AFN.

La construction de la table d'analyse $LR(1)$ est similaire à celle de la table SLR sauf pour les réductions.

> *On ajoute l'action* **réduire** $\boldsymbol{X} \to \alpha$ *à l'entrée table*$[\mathcal{I}, \boldsymbol{a}]$ *uniquement si* $\mathcal{I}$ *contient l'item* $[\boldsymbol{X} \to \alpha\bullet, \boldsymbol{a}]$.

La grammaire est $LR(1)$ si sa table est sans conflit. L'analyse est alors déterministe.

L'analyse $LR(1)$ est plus puissante que les analyses SLR et $LL(1)$.

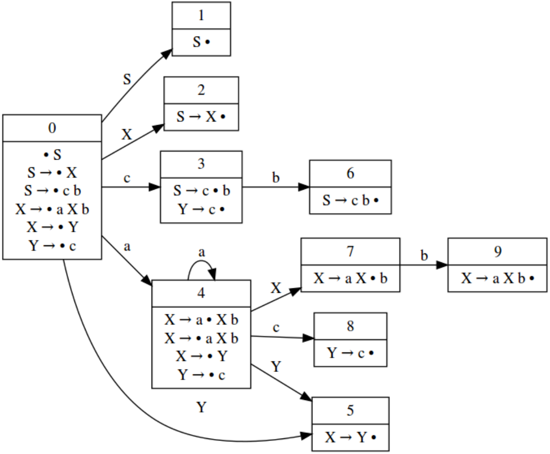
L'inconvénient est qu'on se retrouve avec un très grand nombre d'états. $\leadsto$ Compromis : analyse $LALR$ ...

Analyse $LR(1)$

La grammaire

$$\begin{cases} S & \to & X \mid cb \\ X & \to & aXb \mid Y \\ Y & \to & c \end{cases}$$

n'est pas SLR.



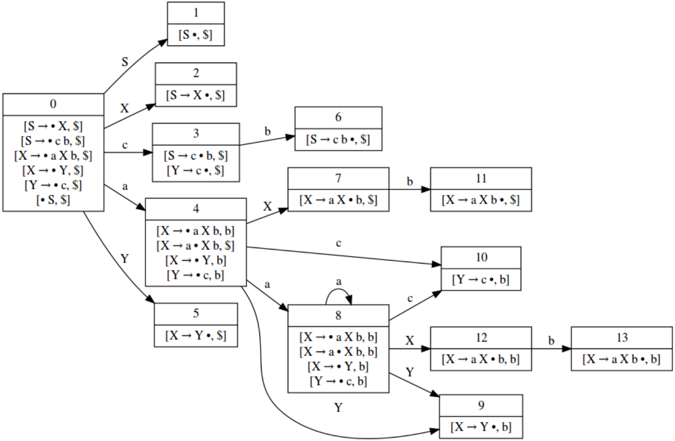
L'automate caractéristique des items $LR(0)$ donné par *Grammophone* présente un conflit décaler/réduire dans l'état 3.

Ce conflit ne peut être tranché avec une analyse SLR car $\boldsymbol{b}$ appartient à Suivant($\boldsymbol{Y}$).

	...	$\boldsymbol{b}$	...
...			
3		**decaler** 6 **reduire** $\boldsymbol{Y \to c}$	

Analyse $\boldsymbol{LR}(1)$

La grammaire $\begin{cases} S & \rightarrow & X \mid cb \\ X & \rightarrow & aXb \mid Y \\ Y & \rightarrow & c \end{cases}$ est $LR(1)$.



L'automate caractéristique des items $LR(1)$ ne présente pas de conflit.

Analyse ***LR***(1)

La table d'analyse $LR(1)$ construite avec *Grammophone* a au plus une action par entrée.

State	c	b	a	$	S	X	Y
0	shift(3)		shift(4)		1	2	5
1				accept			
2				reduce($S \to X$)			
3		shift(6)		reduce($Y \to$ c)			
4	shift(10)		shift(8)			7	9
5				reduce($X \to Y$)			
6				reduce($S \to$ c b)			
7		shift(11)					
8	shift(10)		shift(8)			12	9
9		reduce($X \to Y$)					
10		reduce($Y \to$ c)					
11				reduce($X \to$ a X b)			
12		shift(13)					
13		reduce($X \to$ a X b)					

Analyse LR et ambiguïté

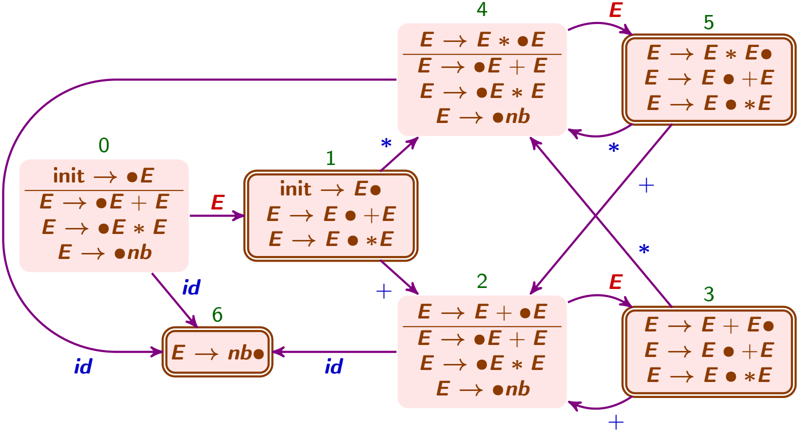
La grammaire $\boldsymbol{E \to E + E \mid E * E \mid nb}$ est ambiguë.
Ce n'est donc pas une grammaire LR.

Cependant, on peut supprimer les conflits décaler/réduire en forçant l'application des règles de priorité et d'associativité des opérateurs.

Pour un conflit $\left|\begin{array}{ll} \textbf{décaler} & B \to Z \bullet\ op1\ T \\ \textbf{réduire} & A \to X\ op2\ Y \bullet \end{array}\right.$

- si $op1$ est prioritaire sur $op2$
  on choisit l'action **décaler** $B \to Z \bullet\ op1\ T$
- si $op2$ est prioritaire sur $op1$
  on choisit l'action **réduire** $A \to X\ op2\ Y \bullet$
- si $op1$ et $op2$ ont même priorité
  - privilégier l'associativité gauche
    $\leadsto$ choisir l'action **réduire** $A \to X\ op2\ Y \bullet$
  - privilégier l'associativité droite
    $\leadsto$ choisir l'action **décaler** $B \to Z \bullet\ op1\ T$

L'automate caractéristique



L'état 1 induit deux conflits décaler/réduire qui se résolvent en inspectant la table des Suivant.

L'état 3 induit deux conflits décaler/réduire sur $+$ et $*$.

L'état 5 induit deux conflits décaler/réduire sur $+$ et $*$.

Supprimer les conflits

	$	+	*	*nb*	*E*
0				**d** 6	1
1	accepter	**d** 2	**d** 4		
2				**d** 6	3
3	**r** $E \rightarrow E + E$	**r** $E \rightarrow E + E$ **d** 2	**r** $E \rightarrow E + E$ **d 4**		
4				**d** 6	5
5	**r** $E \rightarrow E * E$	**r** $E \rightarrow E * E$ **d** 2	**r** $E \rightarrow E * E$ **d** 4		
6	**r** $E \rightarrow nb$	**r** $E \rightarrow nb$	**r** $E \rightarrow nb$		

Lever les conflits et rendre l'analyse déterministe en appliquant les règles usuelles de priorité et d'associativité :

- **priorité de la multiplication sur l'addition**
- **associativité gauche de l'addition**
- **associativité gauche de la multiplication**