

# 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**.

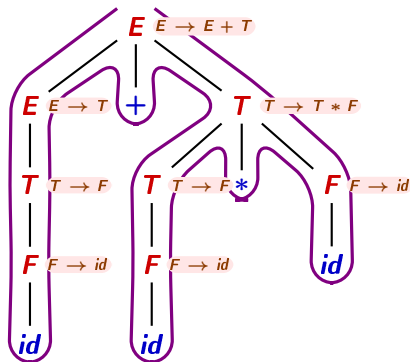
# Analyse syntaxique ascendante

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

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

La suite des décalages et des réductions dans l'ordre du parcours postfixé donne l'arbre de dérivation.

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



Analyse du mot  $id + id * id$

# 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  $\beta$  correspondant à la partie droite d'une production  $A \rightarrow \beta$  qu'on remplace alors par  $A$

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

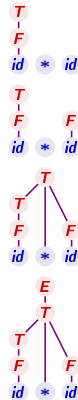
# Analyse syntaxique ascendante

## Approche non-déterministe

Analyse du mot *id \* id* en devinant les bonnes opérations.

Mot	Pile	Arbre	
<i>id * id</i> \$	\$	decaler	<i>id</i>
<i>id * id</i> \$	\$ <i>id</i>	reduire $F \rightarrow id$	<i>F</i>   <i>id</i>
<i>id * id</i> \$	\$ <i>F</i>	reduire $T \rightarrow F$	<i>T</i>   <i>F</i>   <i>id</i>
<i>id * id</i> \$	\$ <i>T</i>	decaler	<i>T</i>   <i>F</i>   <i>id</i> *
<i>id</i> * <i>id</i> \$	\$ <i>T</i> *	decaler	<i>T</i>   <i>F</i>   <i>id</i> * <i>id</i>

$\vdots$	$\vdots$	
$id * id \$$	$\$ T *$	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 va construire 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**  $\rightarrow S$ .

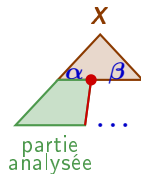
**init** est alors le nouvel axiome et remplace l'ancien axiome  $S$ .

# Les éléments de l'automate

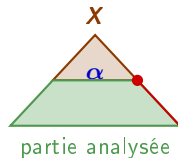
## Item $LR(0)$

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

*ce qui a déjà été analysé • ce qu'on attend*



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



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



# L'automate fini caractéristique

L'AFN avec  $\varepsilon$ -transitions qui caractérise les items valides :

l'alphabet :  $\Sigma \cup N$

les états : les items  $LR(0)$

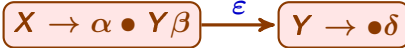
les états d'acceptation : les items  $LR(0)$  complets

l'état initial : l'item  $\text{init} \rightarrow \bullet S$

la fonction de transition : trois types de transition

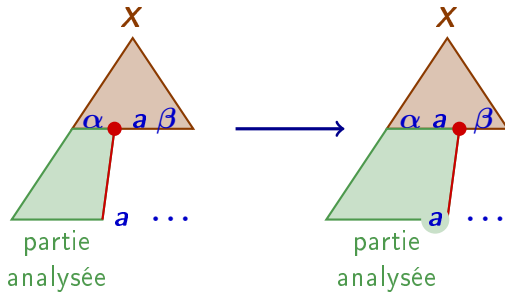
- décalage 

- réduction 

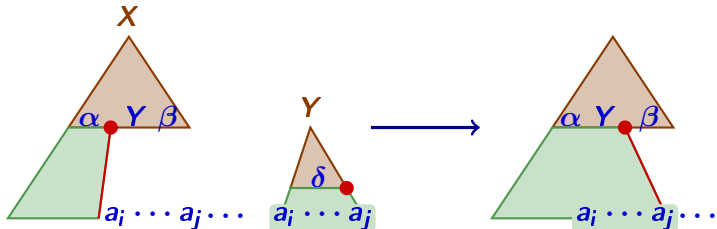
- prédiction   
pour toute production  $Y \rightarrow \delta$

# Les transitions

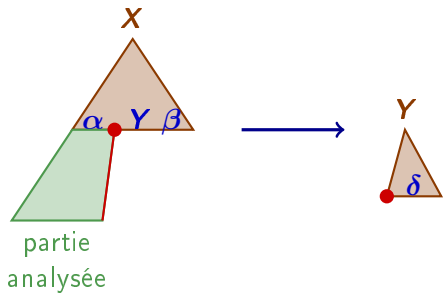
- décalage



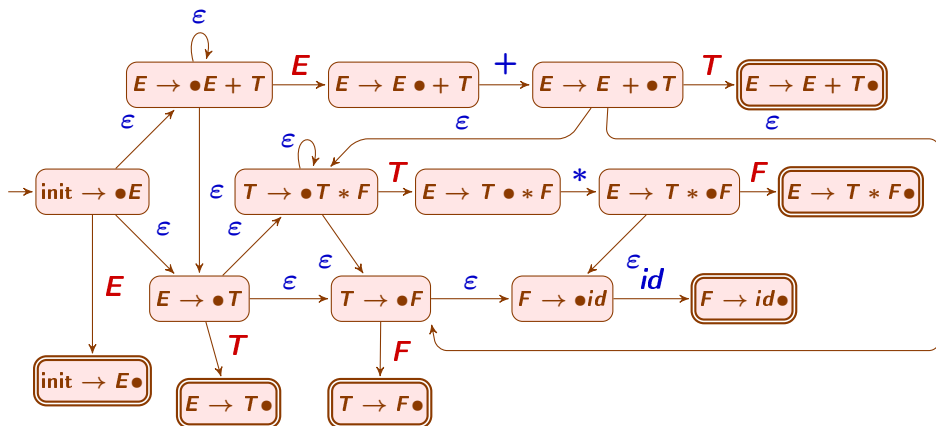
- réduction



- prédiction



# l'AFN avec $\epsilon$ -transitions



La grammaire augmentée

$$\left\{ \begin{array}{ll} \text{init} & \rightarrow E \\ E & \rightarrow E + T \mid T \\ T & \rightarrow T * F \mid F \\ F & \rightarrow id \end{array} \right.$$

## Détermination de l'AFN avec $\varepsilon$ -transitions

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

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

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

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

$$\delta(\mathcal{I}, e) = \text{Cloture}(\{X \rightarrow \alpha e \bullet \beta : X \rightarrow \alpha \bullet e \beta \in \mathcal{I}\})$$

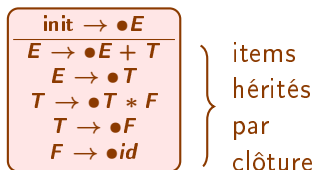
L'état initial est **Cloture**( $\{\text{init} \rightarrow \bullet S\}$ ).

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

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

# Les états de l'AFD

0 état initial



1 =  $\delta(0, E)$



2 =  $\delta(0, T)$



3 =  $\delta(0, F)$



4 =  $\delta(0, id)$



$$5 = \delta(1, +)$$

$$\left\{ \begin{array}{l} E \rightarrow E + \bullet T \\ T \rightarrow \bullet T * F \\ T \rightarrow \bullet F \\ F \rightarrow \bullet id \end{array} \right\} \text{cl\^oture}$$

$$6 = \delta(2, *)$$

$$\left\{ \begin{array}{l} T \rightarrow T * \bullet F \\ F \rightarrow \bullet id \end{array} \right\} \text{cl\^oture}$$

$$7 = \delta(5, T)$$

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

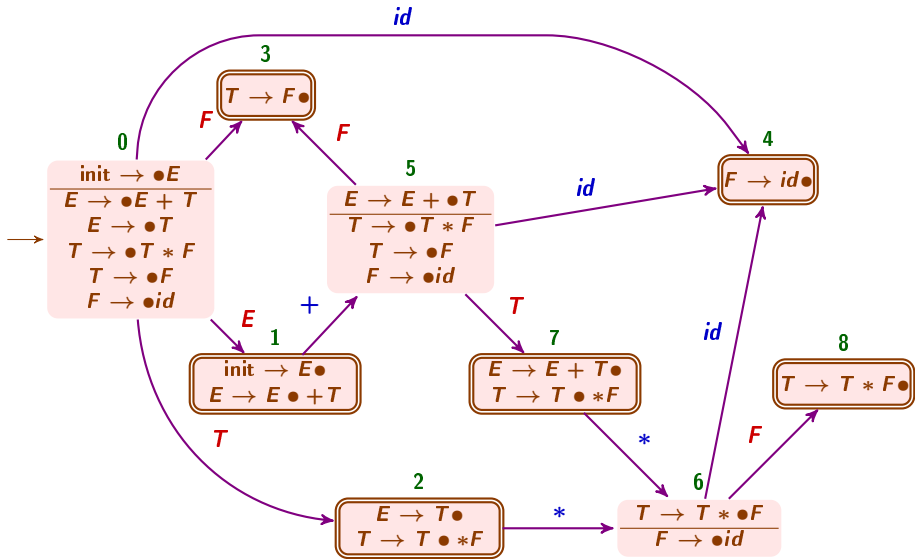
$$\delta(5, F) = 3, \delta(5, id) = 4$$

$$8 = \delta(6, F)$$

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

$$\delta(6, id) = 4$$

# L'AFD





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

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 :

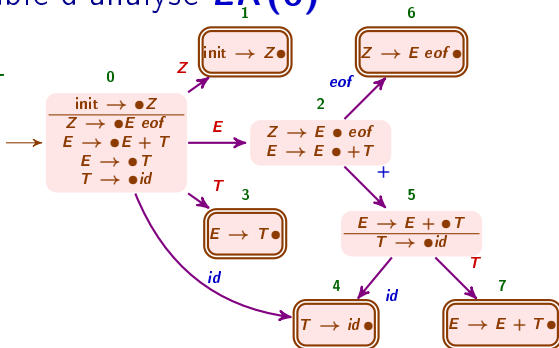
- pour chaque transition  $\mathcal{I} \xrightarrow{a} \mathcal{J}$  étiquetée par un terminal  $a$  ajouter l'action **décaler**  $\mathcal{J}$  à l'entrée  $\text{table}[\mathcal{I}, a]$
- pour chaque transition  $\mathcal{I} \xrightarrow{X} \mathcal{J}$  étiquetée par une variable  $X$ , enregistrer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

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

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

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

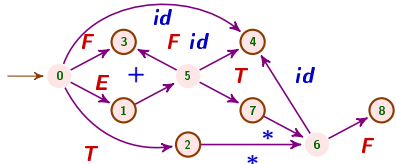
$$\begin{cases} Z \rightarrow E \text{ eof} \\ E \rightarrow E + T \mid T \\ T \rightarrow id \end{cases}$$



	\$	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 $Z \rightarrow E \text{ eof}$	r $Z \rightarrow E \text{ eof}$	r $Z \rightarrow E \text{ eof}$	r $Z \rightarrow E \text{ eof}$				
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 $LR(0)$

$$\left\{ \begin{array}{l} \text{init} \rightarrow E \\ E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow id \end{array} \right.$$



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

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

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

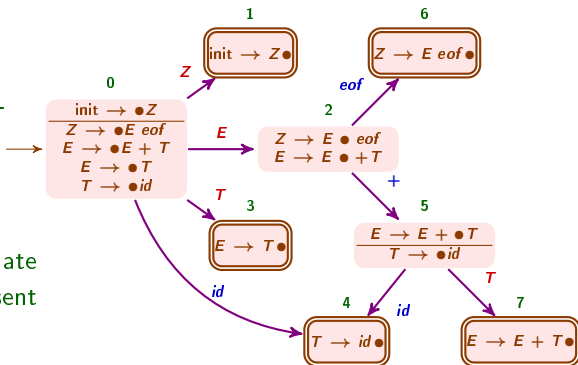
# Grammaire $LR(0)$

La grammaire

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

est  $LR(0)$ .

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



	\$	eof	+	id	E	T	Z
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 $Z \rightarrow E \text{ eof}$	r $Z \rightarrow E \text{ eof}$	r $Z \rightarrow E \text{ eof}$	r $Z \rightarrow E \text{ eof}$			
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.

init  $\rightarrow E \bullet$   
 $E \rightarrow E \bullet + T$

$E \rightarrow T \bullet$   
 $T \rightarrow T \bullet * F$

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

	\$	+	*	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  $table[q, c] = \text{décaler } r$  alors  
empiler  $c$  puis  $r$ , et avancer dans la lecture du mot analysé
- Si  $table[q, c] = \text{réduire } X \rightarrow \alpha$  alors  
dépiler  $2|\alpha|$  symboles, étant donné  $r$  le nouvel état au sommet de la pile empiler  $X$  puis  $\delta(r, X)$ , et afficher  $X \rightarrow \alpha$
- Si  $table[q, c] = \text{accepter}$  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 $E \rightarrow T$	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 $E \rightarrow E + T$			

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

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

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

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

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

$$\begin{array}{l} E \rightarrow E + T \bullet \\ T \rightarrow T \bullet * F \end{array}$$

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

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

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

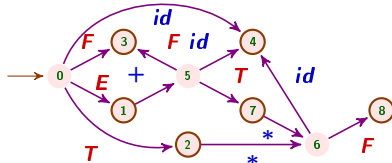
Pour chaque état  $\mathcal{I}$  de l'automate :

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

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

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

# 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 $E \rightarrow T$	r $E \rightarrow T$	d 6				
3	r $T \rightarrow F$	r $T \rightarrow F$	r $T \rightarrow F$				
4	r $F \rightarrow id$	r $F \rightarrow id$	r $F \rightarrow id$				
5				d 4		7	3
6				d 4			8
7	r $E \rightarrow E + T$	r $E \rightarrow E + T$	d 6				
8	r $T \rightarrow T * F$	r $T \rightarrow T * F$	r $T \rightarrow T * F$				

# Analyseur *SLR*

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

Pour examiner un mot, l'analyseur *SLR* utilise la table d'analyse *SLR* et une pile.

Initialement la pile contient l'état initial **0**.

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

# Analyseur *SLR*

	\$	+	*	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
<i>id</i> + <i>id</i> * <i>id</i> \$	0	decaler 4
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>id</i> 4	reduire $F \rightarrow id$ $\delta(0, F) = 3$
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>F</i> 3	reduire $T \rightarrow F$ $\delta(0, T) = 2$
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>T</i> 2	reduire $E \rightarrow T$ $\delta(0, E) = 1$
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1	decaler 5
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1 + 5	decaler 4
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1 + 5 <i>id</i> 4	reduire $F \rightarrow id$ $\delta(5, F) = 3$

# Analyseur *SLR*

⋮	⋮
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1 + 5 <i>id</i> 4
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1 + 5 <i>F</i> 3
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1 + 5 <i>T</i> 7
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1 + 5 <i>T</i> 7 * 6
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1 + 5 <i>T</i> 7 * 6 <i>id</i> 4
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1 + 5 <i>T</i> 7 * 6 <i>F</i> 8
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1 + 5 <i>T</i> 7
<i>id</i> + <i>id</i> * <i>id</i> \$	0 <i>E</i> 1

⋮
reduire <i>F</i> → <i>id</i> $\delta(5, F) = 3$
reduire <i>T</i> → <i>F</i> $\delta(5, T) = 7$
decaler 6
decaler 4
reduire <i>F</i> → <i>id</i> $\delta(6, F) = 8$
reduire <i>T</i> → <i>T</i> * <i>F</i> $\delta(5, T) = 7$
reduire <i>E</i> → <i>E</i> + <i>T</i> $\delta(0, E) = 1$
SUCCESS

	\$	+	*	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	<i>E</i>	→	<i>E</i> + <i>T</i>
2	<i>E</i>	→	<i>T</i>
3	<i>T</i>	→	<i>T</i> * <i>F</i>
4	<i>T</i>	→	<i>F</i>
5	<i>F</i>	→	<i>id</i>

# Analyse $LR(1)$

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  $LR(1)$  de la forme

$[A \rightarrow \alpha \bullet \beta, a]$  où  $A \rightarrow \alpha\beta$  une règle de la grammaire et  $a$  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  $[\text{init} \rightarrow S\bullet, \$]$

**Les états d'acceptation** sont ceux avec des items  $[Y \rightarrow \delta\bullet, c]$

**Les transitions** sont définies ainsi

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

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

$$[X \rightarrow \alpha \bullet Y\beta, b] \xrightarrow{\epsilon} [Y \rightarrow \bullet \delta, c]$$

pour toute production  $Y \rightarrow \delta$  et tout  $c$  dans  $\text{Premier}(\beta b)$  31

## Analyse $LR(1)$

On détermine 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**  $X \rightarrow \alpha$  à l'entrée  $table[\mathcal{I}, a]$  uniquement si  $\mathcal{I}$  contient l'item  $[X \rightarrow \alpha\bullet, a]$ .*

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.  $\rightsquigarrow$  Compromis : analyse  $LALR$  ...



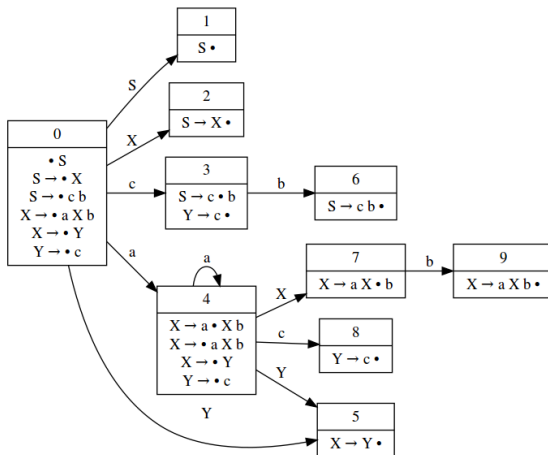
# Analyse $LR(1)$

La grammaire

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

n'est pas  $SLR$ .

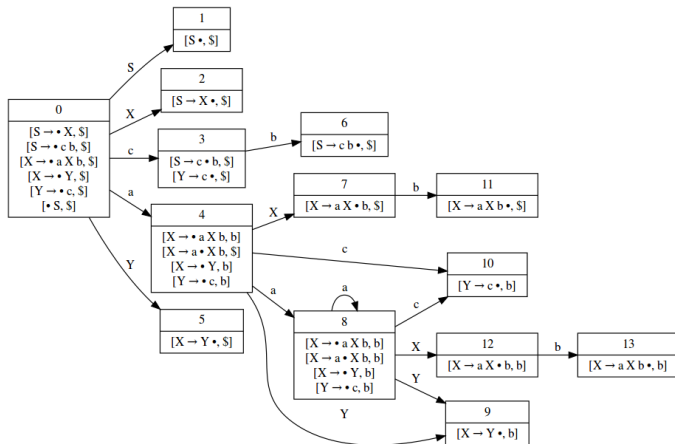
L'automate caractéristique des items  $LR(0)$  donné par *Calgary* présente un conflit décaler/réduire dans l'état 3. Ce conflit ne peut être tranché avec une analyse  $SLR$  car  $b$  appartient à  $Suivant(Y)$ .



	...	$b$	...
...			
3		decaler 6 réduire $Y \rightarrow c$	

# Analyse $LR(1)$

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 *Calgary* 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( $S \rightarrow X$ )			
3		shift(6)		reduce( $Y \rightarrow c$ )			
4	shift(10)		shift(8)			7	9
5				reduce( $X \rightarrow Y$ )			
6				reduce( $S \rightarrow c b$ )			
7		shift(11)					
8	shift(10)		shift(8)			12	9
9		reduce( $X \rightarrow Y$ )					
10		reduce( $Y \rightarrow c$ )					
11				reduce( $X \rightarrow a X b$ )			
12		shift(13)					
13		reduce( $X \rightarrow a X b$ )					

# Analyse LR et ambiguïté

La grammaire  $E \rightarrow E + E \mid E * E \mid nb$  est ambiguë.

Ce n'est donc pas une grammaire LR.

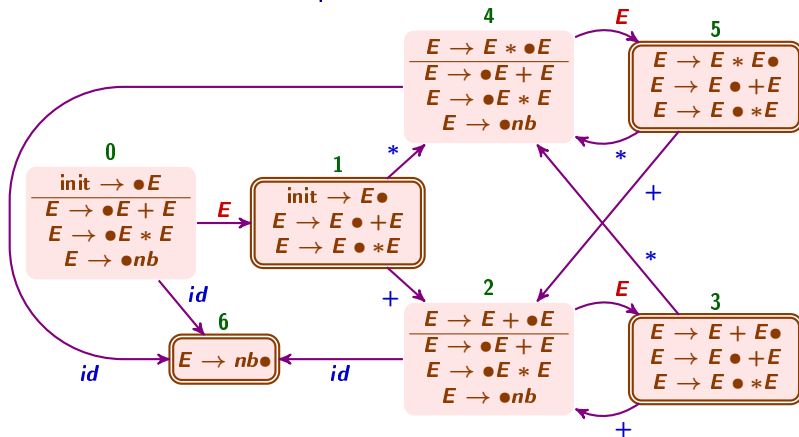
Cependant, on peut supprimer les conflits décaler/réduire en forçant l'application des règles de priorité et d'associativité des opérateurs.

Pour un conflit

décaler	$B \rightarrow Z \bullet op1 T$
réduire	$A \rightarrow X op2 Y \bullet$

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

# 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

	\$	+	*	<i>nb</i>	<i>E</i>
0				d 6	1
1	<b>accepter</b>	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**