Analyse Syntaxique

J.-Cl. Fernandez

September 13, 2004

1 Définitions et notations

Grammaire Soit $G = (V_T \cup \{\$\}, V_N, Z, P)$ (où V_T dénote l'ensemble des symboles terminaux, V_N, l'ensemble des symboles non-terminaux, Z l'axiome et P l'ensemble des règles de production) une grammaire étendue par une règle $Z \longrightarrow S\$$ (le non-terminal Z n'apparaît qu'une seule fois dans les productions). La chaîne à analyser sera terminée par le symbole \$, qui n'appartient pas au vocabulaire terminal. On note $V = V_T \cup V_N$, le vocabulaire de la grammaire.

Automate à pile Un automate à pile $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ est défini par :

- Q est un ensemble fini d'états,
- Σ est un ensemble fini, le vocabulaire,
- Γ est un ensemble fini de symboles de pile,
- q_0 est l'état initial,
- Z_0 est soit ϵ, soit un élément de Γ,
- $F \subseteq Q$ est l'ensemble des états terminaux.
- $\delta : Q \times \Gamma^* \times (\Sigma \cup \{\epsilon\}) \longrightarrow 2^{Q \times \Gamma^*}$ est la fonction de transition.

Une **configuration** est un élément de $Q \times \Gamma^* \times \Sigma^*$. La configuration initiale est (q_0, Z_0, w) où w est la chaîne à lire, une configuration terminale est (q, ϵ, ϵ) (critère d'acceptation "pile vide") ou $(q, \gamma, \epsilon), q \in F)$ (critère d'acceptation "état final"). Une configuration $(q', \alpha\gamma, w')$ est *dérivable* de la configuration $(q, \alpha\beta, uw')$, pour $u \in \Sigma$ ou $u = \epsilon$ si et seulement si $(q', \gamma) \in \delta(q, \beta, u)$. On notera $(q, \alpha\beta, uw') \vdash (q', \alpha\gamma, w')$. Le sommet de pile est à droite dans la chaîne. Le langage engendré par l'automate à pile est :

$L(P) = \{w \in \Sigma^* \mid \exists$ une configuration terminale c vérifiant $(q_0, Z_0, w) \vdash^* c\}$.

Théorème 1.1 *Pour toute gramaire hors-contexte G, il existe une automate à pile P tel que L(G)=L(P) (on a la propriété réciproque).*

On s'intéresse dans la suite à

- la correspondance entre grammaire LR(k) et analyse asscendante.
- la correspondance entre grammaire LL(k) et analyse descendante,

2 Principe de l'analyse ascendante

A une grammaire hors-contexte $G = (V_T, V_N, Z, P)$ on peut associer un automate à pile $P = (Q, \Sigma, \Gamma, \delta, q, Z_0, F)$ où : $Q = \{q, f\}$, $\Sigma = V_T$, $Z_0 = \epsilon$, $F = \emptyset$, $\Gamma = V$,

$$\begin{cases} \delta(q, \epsilon, t) & = & \{(q, t)\} \text{ pour } t \text{ dans } V_T \\ \delta(q, \alpha_1 \cdots \alpha_n, \epsilon) & = & \{(q, A) \mid A \longrightarrow \alpha_1 \cdots \alpha_n \in P\}. \\ \delta(q, Z, \epsilon) & = & \{(f, \epsilon)\} \end{cases}$$

Mais cet automate est non-déterministe. On va montrer comment construire, lorsque c'est possible, un automate déterministe.

2.1 Analyse LR

L'analyseur **LR** utilise une pile sur laquelle il effectue les réductions. La pile contient une chaîne de la forme :

$$q_0 X_1 q_1 X_2 \ldots X_n q_n$$

où : $X_i \in V = V_N \cup V_T$ et q_i est un état. La table d'analyse est divisée en deux :

- une fonction des actions d'analyse : *Action*, Cette fonction détermine, en fonction d'un état et d'un symbole donné une des 4 actions à faire : `Lire`, `Réduire`, `Accepter`, `Erreur`.
- une fonction de transfert : $Trans$. Cette fonction est le calcul des successeurs. On la consulte pour une action de lecture ou de réduction pour déterminer le nouvel état courant.

Un *configuration* est un triplet (état,pile, chaîne d'entrée). Soit la configuration suivante :

$$(q_n, q_0 X_1 q_1 X_2 \ldots X_n q_n, a_i a_{i+1} \ldots a_m)$$

.

- Si $action[q_n, a_i]$ = `Lire` alors soit $q = Trans[q_n, a_i]$. On empile q et la nouvelle configuration est :
$$(q, q_0 X_1 q_1 X_2 \ldots X_n q_n a_i q, a_{i+1} \ldots a_m)$$
.
- Si $action[q_n, a_i]$ = `Réduire` $A \longrightarrow \beta$ alors l'analyseur exécute la réduction :
$$(q_0 X_1 q_1 X_2 \ldots X_{n-r} q_{n-r} A q, a_i a_{i+1} \ldots a_m)$$
où $q = Trans[q_{n-r}, A]$ et $r = \mid \beta \mid$. L'analyseur dépile $2r$ symboles, empile A et l'entrée pour $Trans[q_{n-r}, A]$. Le symbole courant de l'entrée n'est pas changé(on a $X_{n-r+1} \ldots X_n = \beta$).
- `Accepter` : l'analyse est terminée.
- `Erreur` : Appel éventuel d'une procédure de recouvrement.
- la configuration initiale est $(0, 0, \omega)$ où ω est la chaîne à analyser et 0 l'état initial.

Exemple : grammaire des expressions

Nous donnons ci-dessous un automate d'analyse ascendante pour le langage des expressions ainsi que son fonctionnement lors de l'analyse d'une expression.

La méthode de construction de l'automate est décrite dans les paragraphes suivants.

E $\longrightarrow$ E + T(1) E $\longrightarrow$ T (2) F $\longrightarrow$ (E) (5)
T $\longrightarrow$ T * F (3) T $\longrightarrow$ F (4) F $\longrightarrow$ idf (6)

On regroupe *Action* et $Trans$.

Etats	idf	+	*	(	)	$	E	T	F
0	4			5			1	2	3
1		6				acc			
2		r_2	7		r_2	r_2			
3		r_4	r_4		r_4	r_4			
4		r_6	r_6		r_6	r_6			
5	4			5			8	2	3
6	4			5				9	3
7	4			5					10
8		6			11				
9		r_1	7		r_1	r_1			
10		r_3	r_3		r_3	r_3			
11		r_5	r_5		r_5	r_5			

Analyse de la phrase idf * idf + idf$:

0	idf * idf + idf $
0 idf 4	* idf + idf $
0 F 3	* idf + idf $
0 T 2	* idf + idf $
0 T 2 * 7	idf + idf $
0 T 2 * 7 idf 4	+ idf $
0 T 2 * 7 F 10	+ idf $
0 T 2	+ idf $
0 E 1	+ idf $
0 E 1 + 6	idf $
0 E 1 + 6 idf 4	$
0 E 1 + 6 F 3	$
0 E 1 + 6 T 9	$
0 E 1	$
acc	

2.2 Analyse LR(0)

Definition 2.1 *Une situation LR(0) est de la forme* $[X \longrightarrow \mu.\nu]$ *où* $X \longrightarrow \mu\nu$ *est une production de* P.

Le point dans la partie droite de la règle d'une situation LR(0) indique jusqu'où un contexte a été reconnu. Par exemple, un point en fin de règle indique qu'une réduction est possible. Un état d'un

automate déterministe pour un analyseur LR(0) est un ensemble de situations LR(0). Les états et les transitions de l'automate d'analyse sont déterminés par les algorithmes ci-dessous.

Definition 2.2 *La fermeture d'un ensemble de situations LR(0) est (le plus petit ensemble) déterminée de la manière suivante:*

$\textbf{Fermeture}(M) = M \cup \{[X \longrightarrow .\alpha] \mid \exists [Y \longrightarrow \mu.X\gamma] \in \textbf{Fermeture}(M) \text{ et } X \longrightarrow \alpha \in P\}$

Algorithme de construction des états de l'automate

q, q', q_0 sont des états de l'automate
Q est l'ensemble des états de l'automate
$\nu, \gamma, \ldots \in V^*$
$v \in V$

$q_0 = \textbf{Fermeture}(\{[Z \longrightarrow .S]\}); Q = \{q_0\}$ (1)
tant qu'il existe un élément de Q non considéré faire (2)
  soit $q \in Q$ (3)
  soit $\text{base}(q, v) = \{[X \longrightarrow \mu v.\gamma] \mid [X \longrightarrow \mu.v\gamma] \in q\}$ (4)
  si $\text{base}(q, v) \neq \emptyset$ alors (5)
    $q' = \textbf{Fermeture}(\text{base}(q, v))$; (6)
    $Q = Q \cup \{q'\}$ (7)
(8)

Fonction de transition

$\delta(q, v) = q'$ ssi $\exists [X \longrightarrow \alpha.u\gamma] \in q$ et $[X \longrightarrow \alpha u.\gamma] \in q'$

Actions

$$
\begin{array}{lcll}
Action[q,v] & = & \texttt{Accepter} \text{ si} [Z \longrightarrow S.\$] \in q \text{ et } v = \$ & \text{sinon} \\
Action[q,v] & = & \texttt{Réduire}_{[A \longrightarrow \gamma.]} \; \forall v \in V \;\; \text{si} [A \longrightarrow \gamma.] \in q & \text{sinon} \\
Action[q,v] & = & \texttt{Lire} \text{ si } v \in V_T \text{ et } [A \longrightarrow \alpha.v\omega] \in q & \text{sinon} \\
Action[q,v] & = & \texttt{Erreur} &
\end{array}
$$

Condition LR(0)

L'automate ainsi construit ne doit pas avoir d'état dans lequel il y a les deux possibilités : réduction et lecture d'un symbole en entrée, ou deux réductions différentes (pas de transitions conflictuelles).

Conflit lecture/réduction $\left.\begin{array}{ll} [X \longrightarrow \mu.a\gamma] & (a \in V_T) \\ [Y \longrightarrow \delta.] & \end{array}\right\} \in q$

Conflit réduction/réduction $\left.\begin{array}{l} [X \longrightarrow \gamma.] \\ [Y \longrightarrow \delta.] \end{array}\right\} \in q$

2.3 Analyse SLR(1)

La méthode est basée sur la construction de l'automate LR(0). Dans les états où il y a un conflit (réduction-lecture ou réduction-réduction), on calcule les symboles suivants du non terminal apparaissant en partie gauche de la règle concernée par la réduction afin de lever le conflit. La grammaire est SLR(1) si le conflit peut effectivement être levé :

Le conflit lecture/réduction est levé si $\left.\begin{array}{l}[X \longrightarrow \mu.a\gamma] \quad (a \in V_T) \\ [Y \longrightarrow \delta.]\end{array}\right\} \in q \quad \text{alors} \quad a \notin suivant(Y)$

Le conflit réduction/réduction est levé si De même, si un état comporte un conflit réduction-réduction :

$\left.\begin{array}{l}[X \longrightarrow \gamma.] \\ [Y \longrightarrow \delta.]\end{array}\right\} \in q \quad \text{alors} \quad \texttt{Suivant}(X) \cap \texttt{Suivant}(Y) = \phi$

Automate SLR(1)

On construit l'automate si aucun état n'a de conflit SLR(1) lecture/réduction ou réduction/réduction. La fonction de transition est la même que celle de l'automate LR(0).

Actions

$$\begin{array}{lcll} Action[q,v] & = & \texttt{Accepter} \text{ si} [Z \longrightarrow S.\$] \in q \text{ et } v = \$ & \text{sinon} \\ Action[q,v] & = & \texttt{Réduire}_{[A \longrightarrow \gamma.]} \text{ si} [A \longrightarrow \gamma.] \in q \text{ et } v \in \texttt{Suivant}(A) & \text{sinon} \\ Action[q,v] & = & \texttt{Lire} \text{ si } v \in V_T, [A \longrightarrow \alpha.v\omega] \in q & \text{sinon} \\ Action[q,v] & = & \texttt{Erreur} & \end{array}$$

2.4 Premier et Suivant

Calcul de Premier

Soit $t \in V_T, X et N \in V_N, \alpha \in V^*$

$t \in \texttt{Premier}(X)$ si $X \longrightarrow t\alpha \in P$ ou si $X \longrightarrow N\alpha \in P$ et $t \in \texttt{Premier}(N\alpha)$

$t \in \texttt{Premier}(N\alpha)$ si $N \Longrightarrow^* \varepsilon$ et $t \in \texttt{Premier}(\alpha)$

Calcul de Suivant

$$\begin{array}{lcl} \texttt{Suivant}(S) & = & \{\$\} \\ \texttt{Suivant}(X) & = & \bigcup_{Y \longrightarrow \alpha X \beta} \texttt{Premier}(\beta) \cup (\texttt{Suivant}(Y) \textit{ si } \beta \Longrightarrow^* \varepsilon) \end{array}$$

2.5 Analyse LR(1)

Definition 2.3 *Une situation LR(1) est de la forme* $[X \longrightarrow \mu.\nu; \tau]$ *où* $X \longrightarrow \mu\nu$ *est une production de* P *et* $\tau \in \texttt{Suivant}(X)$.

Un automate déterministe pour un analyseur **LR(1)** a pour état un ensemble de situations LR(1). Ces états sont déterminés par les algorithmes ci-dessous.

Definition 2.4 *La fermeture d'un ensemble de situations LR(1) est déterminée de la manière suivante:* **Fermeture**$(M) = M \cup \{[X \longrightarrow .\alpha; \tau] \mid \exists [Y \longrightarrow \mu.X\gamma; \omega] \in$ **Fermeture**(M) *et* $X \longrightarrow \alpha \in P$ *et* $\tau \in$ `Premier`$(\gamma\omega)\}$.

2.6 Algorithme de construction de l'automate (pour LR(1))

On regroupe les situations LR(1) qui ont même noyau LR(0) :
si $[A \longrightarrow \alpha.\gamma; x_1], \cdots, [A \longrightarrow \alpha.\gamma; x_n] \in q$ on écrira $[A \longrightarrow \alpha.\gamma; \{x_1, \cdots, x_n\}]$.

Algorithme

q, q', q_0 sont des états de l'automate
Q est l'ensemble des états de l'automate
$\nu, \gamma, \ldots \in V^*$
$v \in V$

$$q_0 = \textbf{Fermeture}(\{[Z \longrightarrow .S\$; \{\epsilon\}]\}); Q = \{q_0\} \tag{9}$$
$$\text{tant que qu'il existe un élément de } Q \text{ non considéré faire} \tag{10}$$
$$\quad \text{soit } q \in Q \tag{11}$$
$$\quad \text{soit base}(q, v) = \{[X \longrightarrow \mu v.\gamma; \omega] \mid [X \longrightarrow \mu.v\gamma; \omega] \in q\} \tag{12}$$
$$\quad \text{si base}(q, v) \neq \emptyset \text{ alors} \tag{13}$$
$$\quad\quad q' = \textbf{Fermeture}(\text{base}(q, v)); \tag{14}$$
$$\quad\quad Q = Q \cup \{q'\}; \tag{15}$$

Fonction de transition

La fonction de transition est définie de manière analogue à celle de l'automate LR(0).

$$\delta(q, v) = q' \text{ ssi } \exists [X \longrightarrow \alpha.u\gamma; \tau] \in q \text{ et } [X \longrightarrow \alpha u.\gamma; \tau'] \in q'$$

Actions

$$\begin{array}{lcll} Action[q,v] & = & \texttt{Accepter} \text{ si} [Z \longrightarrow S.\$; \{\epsilon\}] \in q \text{ et } v = \$ & \text{sinon} \\ Action[q,v] & = & \texttt{Réduire } \text{si} [A \longrightarrow \gamma.; \Omega] \in q \text{ et } v \in \Omega & \text{sinon} \\ Action[q,v] & = & \texttt{Lire} \text{ si } v \in V_T, [A \longrightarrow \alpha.v\omega; \Omega] \in q & \text{sinon} \\ Action[q,v] & = & \texttt{Erreur} & \end{array}$$

Condition LR(1)

Conflit lecture/réduction $\left.\begin{array}{ll} [X \longrightarrow \mu.a\gamma; \Omega] & (a \in V_T \cap \Omega') \\ [Y \longrightarrow \delta.; \Omega'] & \end{array}\right\} \in q$

Conflit réduction/réduction $\left.\begin{array}{l} [X \longrightarrow \gamma.; \Omega] \\ [Y \longrightarrow \delta.; \Omega'] \\ \Omega \cap \Omega' \neq \emptyset \end{array}\right\} \in q$

2.7 Analyse LALR(1)

Pour construire l'automate LALR(1), on tient compte des informations sur les suivants comme pour la construction de l'automate LR(1). A chaque création d'état, s'il existe un autre état qui a le même noyau que l'état en cours de création alors on fusionne les deux états en considérant l'union des ensembles de suivants.

On appelle noyau d'un état LR(1) la partie de l'ensemble des situations qui correspond à un état de l'automate LR(0), c'est à dire uniquement la partie règle de production.

3 Principe de l'analyse descendante

A une grammaire hors-contexte $G = (V_T, V_N, Z, P)$ on peut associer un automate à pile
$P = (Q, \Sigma, \Gamma, \delta, q, Z_0, F)$ où : $Q = \{q\}, \quad \Sigma = V_T, \quad Z_0 = Z, \quad F = \emptyset, \quad \Gamma = V,$

$$\begin{cases} \delta(q,t,t) & = & \{(q,\epsilon)\} \text{ pour } t \text{ dans } V_T \\ \delta(q,A,\epsilon) & = & \{(q,\alpha_n \cdots \alpha_1) \mid A \longrightarrow \alpha_1 \cdots \alpha_n \in P\}. \end{cases}$$

Cet automate est non déterministe. On cherche une dérivation gauche à partir de Z, qui engendre la chaîne à analyser.

3.1 Analyse LL

L'analyseur **LL** utilise une pile. Les symboles de pile appartiennent soit au vocabulaire terminal de la grammaire soit au vocabulaire non-terminal.

A un stade quelconque de l'analyse, on est en présence de :

- $S \Longrightarrow_g^* uA\beta$ où u est le préfixe de phrase déjà reconnu, A le symbole non-terminal le plus à gauche et $\beta \in V^*$

- et de la phrase $u.\gamma$, où $\gamma \in V_T^*$ est le reste de la chaîne à analyser.

On sélectionne une production $A \longrightarrow \omega$ telle que le préfixe de longueur k (k>0) dérivant de $\omega\beta$ coincide avec le préfixe de longueur k de γ et on se retrouve dans la situation:

- $S \Longrightarrow_g^* u\omega\beta$ où u est le préfixe de phrase déjà reconnu, $\omega.\beta \in V^*$

- et de la phrase $u.\gamma$, où $\gamma \in V_T^*$ est le reste de la chaîne à analyser.

Il s'agit d'une analyse LL(k) ; si k=1, on parle d'analyse LL(1).

3.2 Définitions

Definition 3.1 *Une grammaire $G = (V_T, V_N, Z, P)$ est LL(1) si et seulement si pour chaque paire de productions distinctes $A \longrightarrow \alpha$ et $A \longrightarrow \beta$,* $\texttt{Directeur}(A \longrightarrow \alpha) \cap \texttt{Directeur}(A \longrightarrow \beta) = \phi$ *où*

$$\textbf{Directeur}(A \longrightarrow \omega) = \textbf{Premier}(\omega \textbf{Suivant}(A))$$

3.3 Construction d'un analyseur

Soit (V_T, V_N, Z, P) une grammaire LL(1). On construit un analyseur syntaxique en utilisant une table M[A,a] A $\in V_N$, a$\in V_T$ telle que

- $A \longrightarrow \alpha$ = M[A,a] si $a \in$ `Directeur`$(A \longrightarrow \alpha)$
- Les symboles de pile de l'analyseur sont les éléments de $V_N \cup V_T$.

Le fonctionnement de l'analyseur dépend du symbole en sommet de pile et du symbole en tête de la chaîne restant à analyser. Initialement la pile contient Z et la chaîne à analyser est α. Soit X le symbole en sommet de pile et a le premier symbole de la chaîne restant à analyser. Trois cas sont à examiner :

- si X=a=\$, l'analyse a réussi,
- si X=a $\neq$\$ alors dépiler X ; lire le symbole a dans la chaîne d'entrée
- Si X $\in V_N$
  - si M[A,a] $\ni A \longrightarrow \alpha$ alors dépiler A et empiler α.
  - si M[A,a] est vide alors il y a une erreur.

4 Construction d'un automate LR(0)

Soit la grammaire des expressions à un niveau de priorité définie par les règles suivantes:

1- Z $\longrightarrow$ S\$
2- S $\longrightarrow$ S + A
3- S $\longrightarrow$ A
4- A $\longrightarrow$ (S)
5- A $\longrightarrow$ idf

L'automate d'analyse LR(0) correspondant est :

état q	situations	symbole v	f(q,v)	action
0	Z$\longrightarrow$.S\$	S	1	
	S$\longrightarrow$.S+A	S	1	
	S$\longrightarrow$.A	A	2	
	A$\longrightarrow$.(S)	(	3	`Lire`
	A$\longrightarrow$.idf	idf	4	`Lire`
1	Z$\longrightarrow$S.\$	\$		STOP
	S$\longrightarrow$S.+A	+	5	`Lire`
2	S$\longrightarrow$A.			Red3
3	A$\longrightarrow$(.S)	S	6	
	S$\longrightarrow$.S+A	S	6	
	S$\longrightarrow$.A	A	2	
	A$\longrightarrow$.(S)	(	3	`Lire`
	A$\longrightarrow$.idf	idf	4	`Lire`
4	A$\longrightarrow$idf.			Red5
5	S$\longrightarrow$S+.A	A	7	
	A$\longrightarrow$.(S)	(	3	`Lire`
	A$\longrightarrow$.idf	idf	4	`Lire`
6	A$\longrightarrow$(S.)	)	8	`Lire`
	S$\longrightarrow$S.+A	+	5	`Lire`
7	S$\longrightarrow$S+A.			Red2
8	A$\longrightarrow$(S).			Red4

5 Construction d'un automate SLR(1)

1- Z $\longrightarrow$ A \$ 2- A $\longrightarrow$ a D 3- A $\longrightarrow$ C c
4- D $\longrightarrow$ D d 5- D $\longrightarrow \varepsilon$ 6- C $\longrightarrow$ b

état q	situations	symbole v	f(q,v)	action	
0	Z$\longrightarrow$.A\$	A	1		
	A$\longrightarrow$.aD	a	2	`Lire`	
	A$\longrightarrow$.C c	C	3		
	C$\longrightarrow$.b	b	4	`Lire`	
1	Z$\longrightarrow$A.\$	\$		STOP	
2	A$\longrightarrow$a.D	D	5		
	D$\longrightarrow$.D d	D	5		
	D$\longrightarrow$.			Red(5)	
3	A$\longrightarrow$C.c	c	6	`Lire`	
4	C$\longrightarrow$b.	c		Red(6)	suivant(C)={c}
5	A$\longrightarrow$aD.	\$		Red(2)	suivant(A)={\$}
	D$\longrightarrow$D.d	d	7	`Lire`	
6	A$\longrightarrow$Cc.	\$		Red(3)	suivant(A)={\$}
7	D$\longrightarrow$Dd.	\$,d		Red(4)	suivant(D)={\$,d}

6 Construction d'un automate LR(1)

1- Z $\longrightarrow$ S \$
2- S $\longrightarrow$ C C
3- C $\longrightarrow$ c C
4- C $\longrightarrow$ d

état q	situations	symbole v	f(q,v)	action
0	[Z$\longrightarrow$.S\$;{ε}]	S	1	
	[S$\longrightarrow$.CC;{\$}]	C	2	
	[C$\longrightarrow$.cC;{c,d}]	c	3	`Lire`
	[C$\longrightarrow$.d;{c,d}]	d	4	`Lire`
1	[Z$\longrightarrow$S.\$;{ε}]	\$		STOP
2	[S$\longrightarrow$C.C;{\$}]	C	5	
	[C$\longrightarrow$.cC;{\$}]	c	6	`Lire`
	[C$\longrightarrow$.d;{\$}]	d	7	`Lire`
3	[C$\longrightarrow$c.C;{c,d}]	C	8	
	[C$\longrightarrow$.cC;{c,d}]	c	3	`Lire`
	[C$\longrightarrow$.d;{c,d}]	d	4	`Lire`
4	[C$\longrightarrow$d.;{c,d}]	c,d		Red4
5	[C$\longrightarrow$CC.;{\$}]	\$		Red2
6	[C$\longrightarrow$c.C;{\$}]	C	9	
	[C$\longrightarrow$.cC;{\$}]	c	6	`Lire`
	[C$\longrightarrow$.d;{\$}]	d	7	`Lire`
7	[C$\longrightarrow$d.;{\$}]	\$		Red4
8	[C$\longrightarrow$cC.;{c,d}]	c,d		Red3
9	[C$\longrightarrow$cC.;{\$}]	\$		Red3

7 Construction d'un automate LALR(1)

Reprenons l'exemple de l'automate LR(1) correspondant à la grammaire suivante (c f. 6) :

1- Z $\longrightarrow$ S \$
2- S $\longrightarrow$ C C
3- C $\longrightarrow$ c C
4- C $\longrightarrow$ d

Les états 3 et 6 peuvent être fusionnés ; leur noyau commun est :
C$\longrightarrow$c.C
C$\longrightarrow$.cC
C$\longrightarrow$.d
De même pour les états 4 et 7, ainsi que 8 et 9.

état q	situations	symbole v	f(q,v)	action
0	[Z$\longrightarrow$.S\$;{ε}]	S	1	
	[S$\longrightarrow$.CC;{\$}]	C	2	
	[C$\longrightarrow$.cC;{c,d}]	c	3	`Lire`
	[C$\longrightarrow$.d;{c,d}]	d	4	`Lire`
1	[Z$\longrightarrow$S.\$;{ε}]	\$		STOP
2	[S$\longrightarrow$C.C;{\$}]	C	5	
	[C$\longrightarrow$.cC;{\$}]	c	3	`Lire`
	[C$\longrightarrow$.d;{\$}]	d	4	`Lire`
3	[C$\longrightarrow$c.C;{c,d,\$}]	C	8	
	[C$\longrightarrow$.cC;{c,d,\$}]	c	3	`Lire`
	[C$\longrightarrow$.d;{c,d,\$}]	d	4	`Lire`
4	[C$\longrightarrow$d.;{c,d,\$}]	c,d,\$		Red4
5	[C$\longrightarrow$CC.;{\$}]	\$		Red2
8	[C$\longrightarrow$cC.;{c,d,\$}]	c,d,\$		Red3

8 Construction d'un automate LL(1)

L=$\{1^n 0^m \mid 0 \leq n < m\}$

1. Soit G1 la grammaire étendue engendrant ce langage dont les productions sont données par :

   $Z \longrightarrow S\$$
   $S \longrightarrow 1S0$
   $S \longrightarrow S0$
   $S \longrightarrow 0$

   Cette grammaire n'est pas LL(1). En effet, `Directeur`(S $\longrightarrow$S 0) $\cap$ `Directeur`(S $\longrightarrow$0) =$\{0\}$

2. Supprimons la récursivité à gauche. On obtient :

   $Z \longrightarrow S\$$
   $S \longrightarrow 1S0$
   $S \longrightarrow 0X$
   $X \longrightarrow 0X$
   $X \longrightarrow \epsilon$

   Cette grammaire n'est pas LL(1). En effet, `Directeur`$(X \longrightarrow 0X) \cap$ `Directeur`$(X \longrightarrow \epsilon) = \{0\}$

3. On modifie la grammaire pour reconnaître le langage L=$\{1^n 0^n 0^m \mid 0 \leq n$ et $0 < m\}$

   $Z \longrightarrow S\$$
   $S \longrightarrow AB$
   $A \longrightarrow 1A0 \mid \epsilon$
   $B \longrightarrow 0Y$
   $Y \longrightarrow B \mid \epsilon$

   Cette grammaire est LL(1).

   `Directeur`(A $\longrightarrow$1 A 0 \$)= { 1}
   `Directeur`(A $\longrightarrow\epsilon$) = `Suivant`(A)= $\{0\} \cup$ `Premier`(B)=$\{0\}$.
   `Directeur`(Y $\longrightarrow$B) = `Premier`(B)=$\{0\}$
   `Directeur`(Y $\longrightarrow\epsilon$) = `Suivant`(Y) = `Suivant`(B) = $\{\$\}$.

9 Relation entre LL(1),LR(0),SLR(1), LALR(1),LR(1)

- $LR(0) \subset SLR(1) \subset LALR(1) \subset LR(1)$

- $LL(1) \subset LR(1)$

10 Comparaison LR -LL

10.1 Grammaire LL(1) et non SLR(1)

Z	$\longrightarrow$	S$
S	$\longrightarrow$	AaAb | BbBa
A	$\longrightarrow$	ϵ
B	$\longrightarrow$	ϵ

10.2 Grammaire SLR(1) et non LL(1)

(rcursivit gauche)

Z	$\longrightarrow$	S$
S	$\longrightarrow$	S a | a

10.3 Grammaire LL(1) et non LALR(1)

Z	$\longrightarrow$	S$
S	$\longrightarrow$	aF | bG
F	$\longrightarrow$	Xc | Yd
G	$\longrightarrow$	Xd | Yc
X	$\longrightarrow$	IA
I	$\longrightarrow$	ϵ
Y	$\longrightarrow$	IB
A	$\longrightarrow$	ϵ
B	$\longrightarrow$	ϵ

11 Relations entre LR

11.1 Grammaire LALR(1) et non SLR(1)

Z $\longrightarrow$ S\$
S $\longrightarrow$ Aa | bAc | dc | bda
A $\longrightarrow$ d

11.2 Grammaire LR(1) et non LALR(1)

Z $\longrightarrow$ S\$
S $\longrightarrow$ Aa | bAc | Bc | bBa
A $\longrightarrow$ d
B $\longrightarrow$ d