Análisis Sintáctico Ascendente

➢ Pivote.- Dada una gramática incontextual $G = (N, \Sigma, P, S)$ y dada una forma sentencial a derechas $\alpha\beta\omega \in (N \cup \Sigma)^*$. Un *pivote* de $\alpha\beta\omega$ es (r, j), donde $r : (A \to \beta) \in P$ y $j =\mid \alpha \mid \geq 0$ (con $A \in N; \alpha, \beta \in (N \cup \Sigma)^*; \omega \in \Sigma^*$), si:

$$S \underset{d}{\overset{*}{\Rightarrow}} \alpha A \omega \underset{d}{\Rightarrow} \alpha\beta\omega \qquad \text{con} \quad r : (A \to \beta) \quad \text{y} \quad j =\mid \alpha \mid$$

➢ Gramática LR(K):

Dado un $k \geq 0$ y una gramática incontextual reducida $G = (N, \Sigma, P, S)$ (el axioma no puede aparecer en ninguna parte derecha de ninguna regla de G), G es *LR(K)* si, con $\gamma, \alpha, \alpha' \in (N \cup \Sigma)^*; \omega, \omega' \in \Sigma^*; A, A' \in N$ se cumple:

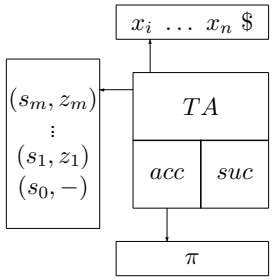
1) $S \underset{d}{\overset{*}{\Rightarrow}} \alpha A \omega \underset{d}{\Rightarrow} \alpha\beta\omega = \gamma\omega \qquad (A \to \beta, \mid \alpha \mid)$,

2) $S \underset{d}{\overset{*}{\Rightarrow}} \alpha' A' \omega' \underset{d}{\Rightarrow} \alpha'\beta'\omega' = \gamma\omega' \qquad (A' \to \beta', \mid \alpha' \mid)$,

3) $\texttt{PRI}_k(\omega) = \texttt{PRI}_k(\omega')$,

entonces $(A \to \beta, \mid \alpha \mid) = (A' \to \beta', \mid \alpha' \mid)$.

Analizador sintáctico ascendente



$z_j \in (N \cup \Sigma); \quad s_j \in Q; \quad x_j \in \Sigma$

$(s_0 \ z_1 \ s_1 \ \ldots z_m \ s_m, \ x_i \ \ldots \ x_n \ \$, \ \pi)$

$\vdash (s_0 \ z_1 \ s_1 \ \ldots z_m \ s_m \ x_i \ s, \ x_{i+1} \ \ldots \ x_n \ \$, \ \pi)$

sii $accion[s_m, \ x_i] = \texttt{desplazar}; \ sucesor[s_m, \ x_i] = s$

o

$\vdash (s_0 \ z_1 \ s_1 \ \ldots z_{m-r} \ s_{m-r} \ A \ s', \ x_i \ \ldots \ x_n \ \$, \ \pi \cdot k)$

sii $accion[s_m, \ x_i] = \texttt{reducir} \quad k : A \to \beta; \quad r = |\beta|;$

$sucesor[s_{m-r}, \ A] = s'$

Ejemplo

	Acción						Sucesor							
	id	+	*	(	)	$	id	+	*	(	)	E	T	F
0	d			d			5			4		1	2	3
1		d				ac		6						
2		r-2	d		r-2	r-2			7					
3		r-4	r-4		r-4	r-4								
4	d			d			5			4		8	2	3
5		r-6	r-6		r-6	r-6								
6	d			d			5			4			9	3
7	d			d			5			4				10
8		d			d			6			11			
9		r-1	d		r-1	r-1			7					
10		r-3	r-3		r-3	r-3								
11		r-5	r-5		r-5	r-5								

0) E' ::= E
1) E ::= E + T
2) E ::= T
3) T ::= T * F
4) T ::= F
5) F ::= (E)
6) F ::= id

	Acción						Sucesor		
	id	+	*	(	)	$	E	T	F
0	d-5			d-4			1	2	3
1		d-6				ac			
2		r-2	d-7		r-2	r-2			
3		r-4	r-4		r-4	r-4			
4	d-5			d-4			8	2	3
5		r-6	r-6		r-6	r-6			
6	d-5			d-4				9	3
7	d-5			d-4					10
8		d-6			d-11				
9		r-1	d-7		r-1	r-1			
10		r-3	r-3		r-3	r-3			
11		r-5	r-5		r-5	r-5			

ASA: Desplazamiento-Reducción

algorithm ASA: Desplazamiento-Reducción
input $G' = (N, \Sigma, P', S); \quad \omega \in \Sigma^*; \quad TA(accion, \ sucesor)$
$accion : Q \times (\Sigma \cup \{\$\}) \to \{desplazar\text{-}s, reducir\text{-}k, aceptar, error\}$
$sucesor : Q \times N \to Q \cup \{error\}$
output **if** $\omega \in L(G)$ **then** π **else** $MenError(\cdot)$;

$\text{push}((s_0, -)); \quad sym = \text{getsym}; \quad \pi = \epsilon; \quad ok = \text{FALSE};$
repeat
  switch $accion[top, sim]$ **do**
  case $desplazar\text{-}s$: $\text{push}((sym, s)); \quad sym = \text{getsym};$
  case $reducir\text{-}k : A \to \beta$
    for $i = 1$ **to** $2 * \mid \beta \mid$ **do** pop;
    if $sucesor[top, A] == error$ **then** $MenError(\cdot)$
    else $s' = sucesor[top, A]; \quad \text{push}((A, s')); \quad \pi = \pi \cdot k$
  case $aceptar$: $ok = \text{TRUE}$
  case $error$: $MenError(\cdot)$;
until ok

Prefijo viable e ítem válido LR(0)

➢ Prefijo viable.- Un *prefijo viable* para una forma sentencial a derechas $\alpha\beta\omega$ (con $\alpha, \beta \in (N \cup \Sigma)^*; \omega \in \Sigma^*$), siendo su pivote asociado $(A \to \beta, \mid \alpha \mid)$ y $\alpha\beta = u_1 \dots u_m$, es cualquier subcadena $u_1 \dots u_i$ con: $0 \le i \le m$ y $u_i \in N \cup \Sigma$.

➢ Teorema de Knuth

El conjunto de todos los prefijos viables de cualquier forma sentencial a derechas de una gramática LR(k), puede ser reconocido por un Autómata de Estados Finitos.

➢ Ítem LR(0).- Sea $G = (N, \Sigma, P, S)$ una gramática incontextual reducida. Un *ítem LR(0)* para G es: $[A \to \beta_1 \cdot \beta_2]$; siendo $(A \to \beta_1\beta_2) \in P$.

➢ Ítem válido LR(0).- Sea $G = (N, \Sigma, P, S)$ una gramática incontextual reducida. Un ítem $[A \to \beta_1 \cdot \beta_2]$ es un *ítem válido LR(0)* para un cierto prefijo viable $(\alpha\beta_1)$, si dado $\alpha, \beta_1, \beta_2 \in (N \cup \Sigma)^*$; $A \in N$; $\omega \in T^*$, cumple que:

$$S \underset{d}{\overset{*}{\Rightarrow}} \alpha\ A\ \omega \underset{d}{\Rightarrow} \alpha\ \beta_1\ \beta_2\ \omega, \qquad \text{siendo el pivote } (A \to \beta_1\ \beta_2, \mid \alpha \mid).$$

LR(0): Cierre y Sucesor

CIERRE(I) Conjunto de ítems LR(0)
given C : Conjunto de ítems LR(0)

$C = I$;
repeat
  for $[A \to \alpha \cdot B\beta] \in C$ **do**
    for $(B \to \gamma) \in P$: $[B \to \cdot\gamma] \notin C$ **do** $C = C \cup \{[B \to \cdot\gamma]\}$;
until no se incorporen nuevos elementos al cierre;
return C;

SUCESOR(I, X) Conjunto de ítems LR(0)
given C : Conjunto de ítems LR(0)

$C = \emptyset$;
for $[A \to \alpha \cdot X\beta] \in I$ **do** $C = C \cup \{[A \to \alpha X \cdot \beta]\}$;
return CIERRE(C);

Colección Canónica de Conjuntos de Ítems LR(0)

Algorithm Colección Canónica de Conjuntos de Ítems LR(0)
input $G = (N, \Sigma, P, S)$;
output $\mathcal{C}$: Colección Canónica de Conjuntos de ítems LR(0);

$\mathcal{C} = \{\text{CIERRE}(\{[\ S' \to \cdot S\ ]\})\}$;
repeat
  for $I \in \mathcal{C}$ **do**
    for $X \in (N' \cup \Sigma)$ **do**
      if $\text{SUCESOR}(I, X) \neq \emptyset \ \wedge\ \text{SUCESOR}(I, X) \notin \mathcal{C}$ **then**
        $\mathcal{C} = \mathcal{C} \cup \text{SUCESOR}(I, X)$;
until no se incorporen nuevos elementos a la colección $\mathcal{C}$;
return $\mathcal{C}$;

Ejemplo: Colección Canónica de Conjuntos de Items LR(0)

0)	E'	→	E	3)	T	→	T * F	5)	F	→	(E)
1)	E	→	E + T	4)	T	→	F	6)	F	→	a
2)	E	→	T								

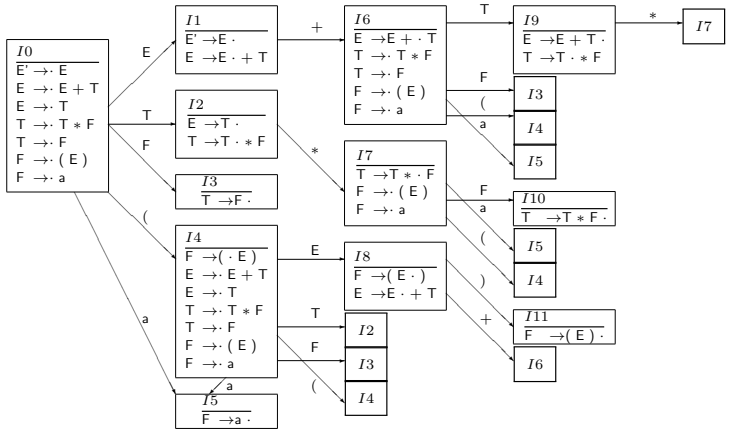


Tabla de Análisis SLR(1)

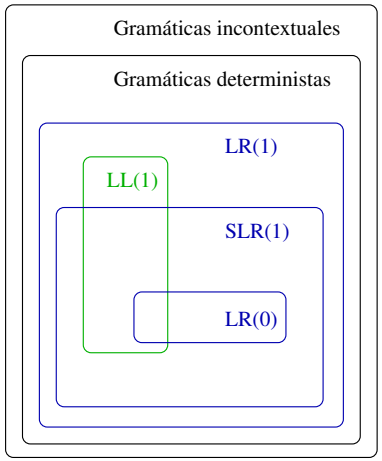
algorithm Construcción de la TA-SLR(1)

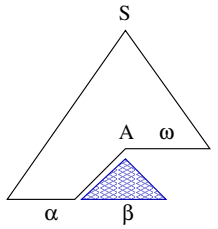
input $G' = (N, \Sigma, P', S)$; y la C.C.C. de ítems LR(0): $\mathcal{C} = \{I_0, I_1, \ldots, I_n\}$;

output TA $(accion,\ sucesor)$ inicializada a $error$;
$accion : Q \times (\Sigma \cup \{\$\}) \rightarrow \{desplazar\text{-}s, reducir\text{-}k, aceptar, error\}$
$sucesor : Q \times N \rightarrow Q \cup \{error\}$

for $I_i \in \mathcal{C}$ $\wedge$ **for** ítem $\mathcal{I} \in I_i$ **do**
  if $(\mathcal{I} == [A \rightarrow \alpha \cdot a\beta] :\ a \in \Sigma)$ $\wedge$ $\text{SUCESOR}(I_i, a) == I_j$ **then**
    $accion[i, a] = desplazar\text{-}j$;
  if $(\mathcal{I} == [A \rightarrow \alpha \cdot B\beta] :\ B \in N')$ $\wedge$ $\text{SUCESOR}(I_i, B) == I_j$ **then**
    $sucesor[i, B] = j$;
  if $\mathcal{I} == [S' \rightarrow S\cdot]$ **then** $accion[i, \$] = aceptar$;
  if $(\mathcal{I} == [A \rightarrow \alpha\cdot])$ $\wedge$ $(k : A \rightarrow \alpha) \in P'$ **then**
    for $a \in$ $\texttt{SIG}(A)$ **do** $accion[i, a] = reducir\text{-}k$;

Relaciones entre gramáticas





- Para gramáticas LL(1) se decide la regla a derivar para A, conociendo α y los $\texttt{PRI}(\beta \cdot \texttt{SIG}(A))$.
- Para gramáticas LR(1) se decide la regla a reducir conociendo $\alpha \cdot \beta$ y los $\texttt{SIG}(A)$.