

ANÁLISIS SINTÁCTICO ASCENDENTE

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

$$S \xrightarrow[d]{*} \alpha A \omega \xRightarrow[d]{*} \alpha \beta \omega \quad \text{con } r : (A \rightarrow \beta) \text{ y } j = |\alpha|$$

➤ **Gramática LR(K):**

Dado un $k \geq 0$ y una gramática incontextual reducida $G = (\Sigma, N, S, P)$ (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) $S \xrightarrow[d]{*} \alpha A \omega \xRightarrow[d]{*} \alpha \beta \omega = \gamma \omega \quad (A \rightarrow \beta, |\alpha|)$,
- 2) $S \xrightarrow[d]{*} \alpha' A' \omega' \xRightarrow[d]{*} \alpha' \beta' \omega' = \gamma \omega' \quad (A' \rightarrow \beta', |\alpha'|)$,
- 3) $PRI_k(\omega) = PRI_k(\omega')$,

entonces $(A \rightarrow \beta, |\alpha|) = (A' \rightarrow \beta', |\alpha'|)$.

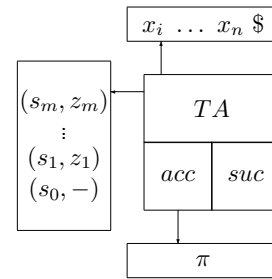
EJEMPLO

	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							
3		r-4		r-4		r-4							
4	d-5			d-4			8	2	3				
5		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							
10		r-3		r-3		r-3							
11		r-5		r-5		r-5							

ANALIZADOR SINTÁCTICO ASCENDENTE



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

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

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

$$\text{si } accion[s_m, x_i] = \text{desplazar}; \text{ sucesor}[s_m, x_i] = s$$

o

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

$$\text{si } accion[s_m, x_i] = \text{reducir } k : A \rightarrow \beta; r = |\beta|;$$

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

ASA: DESPLAZAMIENTO-REDUCCIÓN

algorithm **ASA: Desplazamiento-Reducción**

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

push((s₀, -)); sym = getsym; $\pi = \epsilon$; fin = FALSE;

repeat

switch $accion[top, sym]$ do

case $desplazar-s$: push((sym, s)); sym = getsym;

case $reducir-k$: $A \rightarrow \beta$

for $i = 1$ to $2 * |\beta|$ do pop;

if $sucesor[top, A] == \text{error}$ then $MenError(\cdot)$

else $s' = sucesor[top, A]$; push((A, s')); $\pi = \pi \cdot k$

case $aceptar$: fin = TRUE

case $error$: $MenError(\cdot)$; fin = TRUE

until fin

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 $(A \rightarrow \beta, | \alpha |)$ y $\alpha\beta = u_1 \dots u_m$, es cualquier subcadena $u_1 \dots u_i$ con: $0 \leq i \leq m$ 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 $G = (\Sigma, N, S, P)$ una gramática incontextual reducida.

Un ítem LR(0) para G es: $[A \rightarrow \beta_1 \cdot \beta_2]$; siendo $(A \rightarrow \beta_1 \beta_2) \in P$.

➤ **Ítem válido LR(0).**- Sea $G = (\Sigma, N, S, P)$ $G = (N, \Sigma, P, S)$ una gramática incontextual reducida.

Un ítem $[A \rightarrow \beta_1 \cdot \beta_2]$ 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:

$$S \xRightarrow{*} \alpha A \omega \Rightarrow \alpha \beta_1 \beta_2 \omega, \quad \text{siendo el pivote } (A \rightarrow \beta_1 \beta_2, | \alpha |).$$

LR(0): CIERRE Y SUCESOR

CIERRE(I) Conjunto de ítems LR(0)

given C : Conjunto de ítems LR(0)

$C = I$;

repeat

for $[A \rightarrow \alpha \cdot B\beta] \in C$ **do**

for $(B \rightarrow \gamma) \in P$: $[B \rightarrow \cdot \gamma] \notin C$ **do** $C = C \cup \{[B \rightarrow \cdot \gamma]\}$;

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 $[A \rightarrow \alpha \cdot X\beta] \in I$ **do** $C = C \cup \{[A \rightarrow \alpha X \cdot \beta]\}$;

return CIERRE(C);

COLECCIÓN CANÓNICA DE CONJUNTOS DE ÍTEMS LR(0)

Algorithm Colección Canónica de Conjuntos de Ítems LR(0)

input $G = (\Sigma, N, S, P)$;

output \mathcal{C} : Colección Canónica de Conjuntos de ítems LR(0);

$\mathcal{C} = \{\text{CIERRE}(\{[S' \rightarrow \cdot S]\})\}$;

repeat

for $I \in \mathcal{C}$ **do**

for $X \in (N' \cup \Sigma)$ **do**

if SUCESOR(I, X) $\neq \emptyset \wedge$ 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 ÍTEMS LR(0)

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

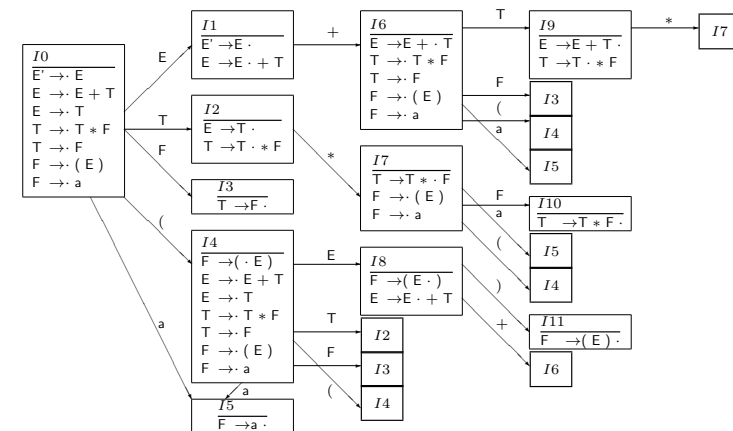


TABLA DE ANÁLISIS SLR(1)

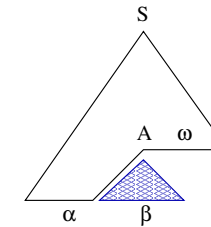
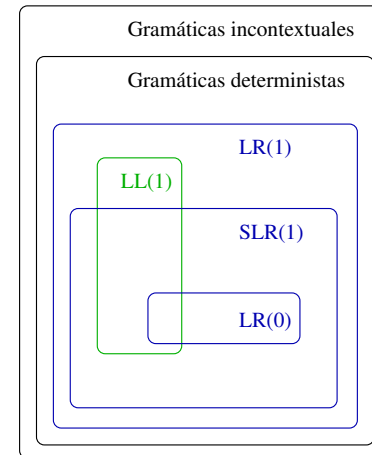
algorithm Construcción de la TA-SLR(1)

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

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

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**
 $accion[i, a] = \text{desplazar-}j$;
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** $accion[i, \$] = \text{aceptar}$;
if $(\mathcal{I} == [A \rightarrow \alpha \cdot] \wedge (k : A \rightarrow \alpha) \in P')$ **then**
for $a \in \text{SIG}(A)$ **do** $accion[i, a] = \text{reducir-}k$;

RELACIONES ENTRE GRAMÁTICAS

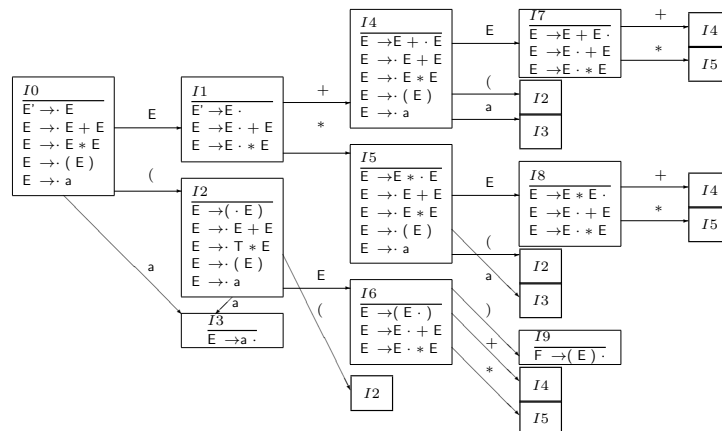


▷ Para gramáticas LL(1) se decide la regla a derivar para A , conociendo α y los $\text{PRI}(\beta \cdot \text{SIG}(A))$.

▷ Para gramáticas LR(1) se decide la regla a reducir conociendo $\alpha \cdot \beta$ y los $\text{SIG}(A)$.

RESOLUCIÓN DE CONFLICTOS: EJEMPLO 1/2

0)	$E' \rightarrow E$	2)	$E \rightarrow E * E$	4)	$E \rightarrow a$
1)	$E \rightarrow E + E$	3)	$E \rightarrow (E)$		



RESOLUCIÓN DE CONFLICTOS: EJEMPLO 2/2

Criterios para resolver conflictos

Sintácticos: Modificar la gramática

$E' \rightarrow E$
 $E \rightarrow E + T \mid T$
 $T \rightarrow T * F \mid F$
 $F \rightarrow (E) \mid a$

Semánticos:

precedencia $(*) >$ precedencia $(+)$,

asociatividad a izquierdas $(*, +)$.

I_7 $accion(I_7, +) = r - 1$ por asociatividad a izquierdas
 $accion(I_7, *) = d - 5$ por mayor precedencia del $*$

I_8 $accion(I_8, +) = r - 2$ por mayor precedencia del $*$
 $accion(I_8, *) = r - 2$ por asociatividad a izquierdas

		Acción						Sucesor
		a	+	*	()	\$	E
0	d-3				d-2			1
1			d-4	d-5			ac	
2	d-3				d-2			6
3		r-4		r-4		r-4	r-4	
4	d-3				d-2			7
5	d-3				d-2			8
6		d-4		d-5		d-9		
7		r-1/d-4		r-1/d-5		r-1	r-1	
8		r-2/d-4		r-2/d-5		r-2	r-2	
9		r-3		r-3		r-3	r-3	

RECUPERACIÓN DE ERRORES

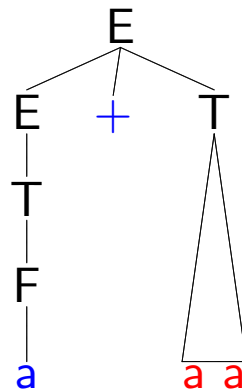
- Un propósito del AS es filtrar todos los errores sintácticos que aparezcan el programa fuente
- Las etapas posteriores del compilador no deberían tratar con construcciones sintácticas mal formadas
- El AS debe identificar los errores para que el programador pueda corregirlos
- No es aceptable que el análisis sintáctico finalice cuando se detecte el primer error
- El AS debe recuperarse lo antes posible para que análisis sintáctico pueda continuar (y detectar otros posibles errores)
- Los errores se detectarán cuando se consulta la tabla de ACCIÓN y encuentre una entrada error. En ningún caso se desplazará un símbolo de entrada erróneo a la pila

RECUPERACIÓN DE ERRORES: MODO PÁNICO

- **Detección del error**, cuando se produzca $ACCIÓN[s_i, a_j] = \text{error}$
- Recuperación de errores en **Modo Pánico**
 1. Buscar en la pila (desapilando) el primer estado s_k para el que exista una transición con un no-terminal: $\exists A \in N: \text{SUCEsor}[s_k, A] = s'$
 2. Apilar ($A \ s'$)
 3. Eliminar símbolos de la entrada hasta que $\exists a \in \Sigma: a \in \text{SIG}(A)$
 4. Restablecer el AS

RECUPERACIÓN DE ERRORES

pila	cad.entrada	acciones
0	a + a a \$	
0 a 5	+ a a \$	reducir-6
0 F 3	+ a a \$	reducir-4
0 T 2	+ a a \$	reducir-2
0 E 1	+ a a \$	
0 E 1 + 6	a a \$	
0 E 1 + 6 a 5	a \$	error
0 E 1 + 6 T 9	\$	reducir-1
0 E 1	\$	aceptar



$E \xRightarrow{r-1} E + T \xRightarrow{error} E + \boxed{a a} \xRightarrow{r-2} T + \boxed{a a} \xRightarrow{r-4} F + \boxed{a a} \xRightarrow{r-6} a + \boxed{a a}$