Análisis Sintáctico Ascendente

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

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

➢ Gramática LR(K):

Dado un $k \geq 0$ y una gramática incontextual reducida $G = (N, T, P, S)$ (el axioma no puede aparecer en ninguna parte derecha de ninguna regla de G), G es $LR(K)$ si, con $\gamma, \alpha, \alpha' \in (N \cup T)^*; \omega, \omega' \in T^*; A, A' \in N$ se cumple:

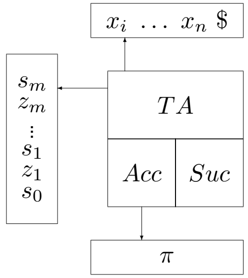
1) $S \overset{*}{\underset{d}{\Rightarrow}} \alpha A \omega \underset{d}{\Rightarrow} \alpha\beta\omega = \gamma\omega \qquad (A \to \beta, \mid \alpha \mid)$,

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

3) $\text{PRIMEROS}_k(\omega) = \text{PRIMEROS}_k(\omega')$,

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

Analizador sintáctico ascendente



$z_j \in (N \cup T) \qquad s_j \in \mathcal{Q} \qquad x_j \in T$

$(s_0 \; z_1 \; s_1 \; \dots z_m \; \boldsymbol{s_m}, \;\; \boldsymbol{x_i} \; \dots \; x_n \; \$, \;\; \pi) \vdash$

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

<u>**sii**</u>

$Acc[s_m, \; x_i] = \texttt{desplazar}$; y $Suc[s_m, \; x_i] = s$

o

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

<u>**sii**</u>

$Acc[s_m, \; x_i] = \texttt{reducir} \; k : A \to \beta$;

$|\beta| = r$; y $Suc[s_{m-r}, \; A] = s'$

$$(s_0, \; \boldsymbol{x} \; \$, \; \epsilon) \overset{*}{\vdash} (s_0 \; S \; s, \; \$, \; \pi)$$

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

Algoritmo: ASA: Desplazamiento-Reducción

Entrada $G' = (N', T, P', S')$; $\omega \in T^*$; TA(acción, sucesor)

acción: $Q \times (T \cup \{\$\}) \to$ {desplazar-s, reducir-k, aceptar, error}

sucesor: $Q \times N \to Q \cup$ { error}

Salida **si** $\omega \in L(G)$ **entonces** π **else** *MenError*()

Método

apilar(s$_0$); sim = *ObtSim*; $\pi = \epsilon$; fin = *falso*;

repetir

caso_de_que acción[cima,sim] **sea**

desplazar-s: *apilar*(sim); *apilar*(s); sim = *ObtSim*;

reducir-k $(A \to \beta)$: **para** $i = 1$ **hasta** $2 * \mid \beta \mid$ **hacer** *desapilar*;

si sucesor[cima,A] = error **entonces** *MenError*().

sino s' = sucesor[cima,A]; *apilar*(A); *apilar*(s'); $\pi = \pi \cdot k$;

aceptar: fin = verdad;

error: *MenError*().

hasta fin

Fin

Prefijo viable e Ítem válido LR(0)

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

➢ 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 $G = (N, T, P, S)$ 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 $G = (N, T, P, S)$ 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 $\alpha, \beta_1, \beta_2 \in (N \cup T)^*$; $A \in N$; $\omega \in T$, cumple que:

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

LR(0): Cierre y Sucesor

Función: CIERRE(I): conjunto de ítems LR(0)
Dado C: conjunto de ítems LR(0)
Método $C := I$;
repetir
  para todo $[A \to \alpha \cdot B\beta] \in C$; **hacer**
    para_todo $(B \to \gamma) \in P$: $[B \to \cdot\gamma] \notin C$ **hacer** $C := C \cup \{[B \to \cdot\gamma]\}$;
hasta no se incorporen nuevos elementos al cierre;
Devolver C;
Fin

Función: SUCESOR(I, X): conjunto de ítems LR(0).
Dado C: conjunto de ítems LR(0).
Método $C := \emptyset$;
para_todo $[A \to \alpha \cdot X\beta] \in I$ **hacer** $C := C \cup \{[A \to \alpha X \cdot \beta]\}$;
Devolver CIERRE(C);
Fin

Colección Canónica de Conjuntos de Ítems LR(0)

Algoritmo: Colección Canónica de Conjuntos de ítems LR(0).
Dado $G' = (N', T, P', S')$;
Salida C: Colección Canónica de Conjuntos de ítems LR(0).
Método
  $C = \{\text{CIERRE}(\{[S' \to \cdot S]\})\}$;
  repetir
    para todo $I \in C$ **hacer**
      para todo $X \in (N' \cup T)$ **hacer**
        si SUCESOR$(I, X) \neq \emptyset \wedge$ SUCESOR$(I, X) \notin C$ **entonces**
          $C = C \cup$ SUCESOR(I, X);
  hasta no se incorporen nuevos elementos a la colección C;
Devolver C;
Fin

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								

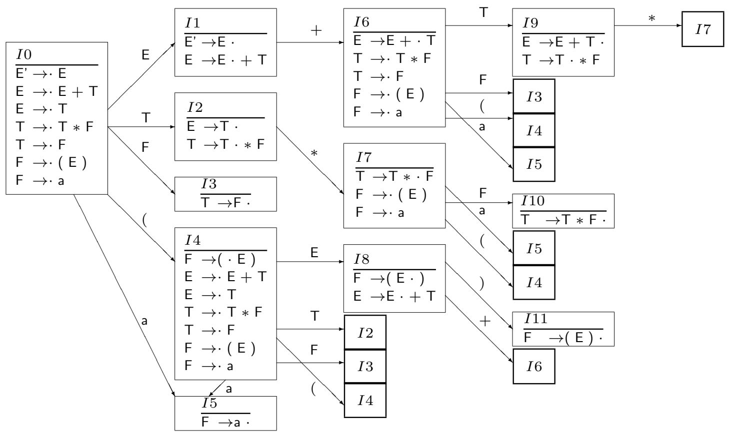


Tabla de Análisis SLR(1)

Algoritmo: Construcción de la TA-SLR(1)

Entrada $G' = (N', T, P', S')$ y la colección LR(0): $C := \{I_0, I_1, \ldots, I_n\}$;

Salida TA (acción, sucesor), inicializada a: `error`;

acción: $Q \times (T \cup \{\$\}) \to$ { `desplazar-s`, `reducir-k`, `aceptar`, `error` },

sucesor: $Q \times N \to Q \cup \{$error$\}$.

Método

para todo $I_i \in C \wedge$ **para_todo** ítem $\Im \in I_i$ **hacer**

si $(\Im = [A \to \alpha \cdot a\beta] : a \in T) \wedge \textsc{sucesor}(I_i, a) = I_j$ **entonces**

acción$[i, a]$ = `desplazar-j`;

si ($\Im = [A \to \alpha \cdot B\beta] : B \in N') \wedge \textsc{sucesor}(I_i, B) = I_j$ **entonces**

sucesor$[i, B] = j$;

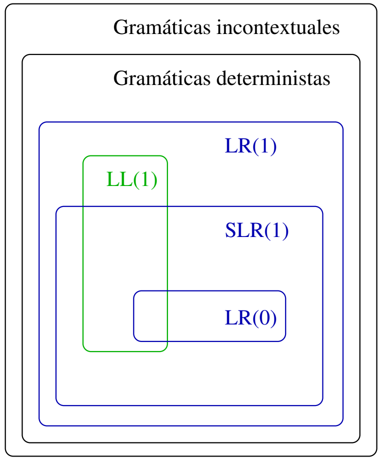
si $\Im = [S' \to S\cdot]$ **entonces** acción$[i, \$]$ = `aceptar`;

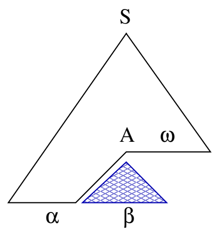
si $(\Im = [A \to \alpha\cdot]) \wedge (k : A \to \alpha) \in P'$ **entonces**

para_todo $a \in \textsc{siguientes}(A)$ **hacer** acción$[i, a]$ = `reducir-k`;

Fin

Relaciones entre gramáticas



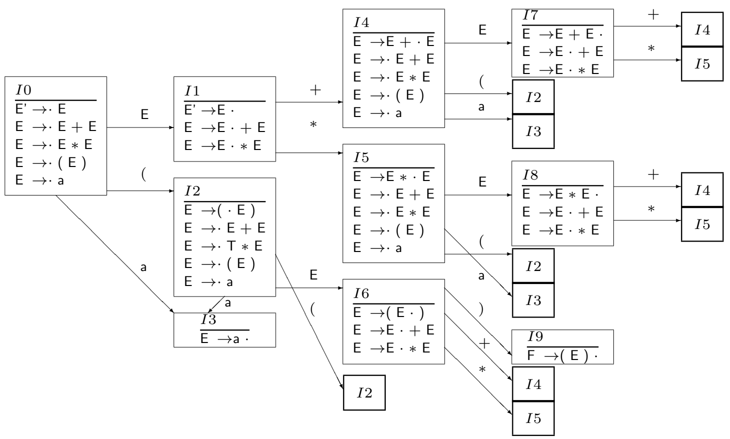


▷ Para gramáticas LL(1) se decide la regla a derivar para A, conociendo α y los $\textsc{primeros}$ $(\beta \cdot \textsc{siguientes}(A))$.

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

Resolución de conflictos: ejemplo 1/2

0)	E'	→	E	2)	E	→	E ∗ E	4)	E	→	a
1)	E	→	E + E	3)	E	→	(E)				



Resolución de conflictos: ejemplo 2/2

Criterios para resolver conflictos

Criterios semánticos:

➢ precedencia $(*) \geq$ precedencia $(*)$,

➢ 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	

I_7	$acción(I_7, +) = r - 1$	por asociatividad a izquierdas
	$acción(I_7, *) = d - 5$	por mayor precedencia del $*$
I_8	$acción(I_8, +) = r - 2$	por mayor precedencia del $*$
	$acción(I_8, *) = r - 2$	por asociatividad a izquierdas

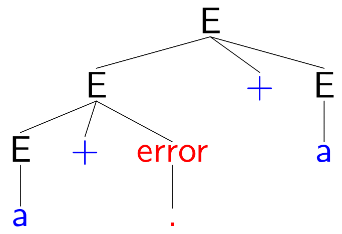
Recuperación de errores

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

- **Detección del error**, cuando se produzca $\text{ACCIÓN}[s_i, a_j] =$ `error`
- Recuperación de errores en **Modo Pánico**
  1. Buscar en la pila (desapilar) el primer estado s_k para el que $\exists A \in N$: $\text{SUCESOR}[s_k, A] = s'$
  2. Eliminar símbolos de la entrada hasta que $\exists a \in T: \; a \in \text{SIGUIENTE}(A)$
  3. Apilar $(A\ s')$
  4. Reestablecer el AS

Recuperación de errores

pila	cad.entrada	acciones
0	a + + a \$	
0 a 3	+ + a \$	`reducir-4`
0 E 1	+ + a \$	
0 E 1 + 4	+ a \$	`error`
0 E 1 + 4 E 7	+ a \$	`reducir-1`
0 E 1	+ a \$	
0 E 1 + 4	a \$	
0 E 1 + 4 a 3	\$	`reducir-4`
0 E 1 + 4 E 7	\$	`reducir-1`
0 E 1	\$	`aceptar`



$$\text{E} \overset{r-1}{\Rightarrow} \text{E + E} \overset{r-4}{\Rightarrow} \text{E + a} \overset{r-1}{\Rightarrow} \text{E + E + a} \overset{error}{\Rightarrow} \text{E + + a} \overset{r-4}{\Rightarrow} \text{a + + a}$$