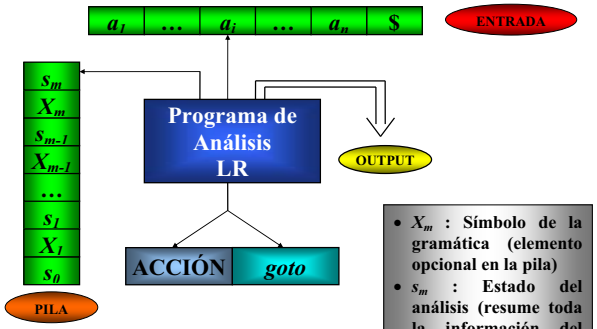
ANÁLISIS ASCENDENTE: ANÁLISIS LR(K).

L: *Left-to-Right Analysis of the Input*.
R: *Rightmost Derivation in Reverse*.
(K): Máximo número de símbolos de la entrada necesarios para tomar una decisión en el analizador sintáctico.



LR(K): VENTAJAS.

- En principio, pueden formalizar cualquier lenguaje de programación generado a partir de gramáticas de contexto libre.
- Es el método de reducción-desplazamiento sin retroceso más **general y eficiente** (es el compromiso perfecto entre eficiencia y generalidad).
- El conjunto de gramáticas *LR(K)* es un superconjunto propio del conjunto *LL(K)*.
- Detecta el error sintáctico tan pronto como esto es posible en la lectura de izqda. a dcha. de la cadena de entrada.

LR(K): INCONVENIENTES.

- Generalmente, es necesario utilizar una herramienta (*YACC*, etc.) para generar el analizador sintáctico, dada la gran complejidad de esta tarea.

1. Conceptos preliminares.

a) Tipos de analizadores LR(K) – determinados por la construcción de sus tablas:
- **i) SLR** (Simple LR)
- **ii) LARL** (LookAhead LR)
- **iii) LR Canónico**

COMPLEJIDAD de IMPLEMENTACIÓN (+) — POTENCIA (+)

b) Prefijo viable:
- **i) Definición formal:** Es una forma sentencial determinada con derivaciones por la derecha que no continúa más allá del extremo derecho del *pivote* situado más a la derecha de esa forma sentencial.
- **ii) Definición informal:** Cada una de las formas sentenciales que pueden leerse sobre la pila del analizador ascendente en un momento dado de su proceso (de abajo a arriba).
- **iii) Observación:** No habrá error en el análisis siempre que la porción examinada de la entrada hasta un punto dado pueda reducirse a un prefijo viable.

c) *Ítem LR*(0) o *ítem* de una gramática G.
- **i) Definición formal:** Es una producción de G con un punto en una cierta posición de su parte derecha.
- **ii) Ejemplo:** La producción $A \rightarrow XYZ$ origina 4 posibles ítems *LR*(0):
  - **a)** $A \rightarrow \bullet XYZ$ **c)** $A \rightarrow XY\bullet Z$
  - **b)** $A \rightarrow X\bullet YZ$ **d)** $A \rightarrow XYZ\bullet$
- **iii) Significado:** Es la porción procesada de la parte derecha de una producción en un instante determinado (lo que queda a la izquierda del punto).
- **iv) Observación:** La producción $A \rightarrow \lambda$ genera un único ítem *LR*(0): $A \rightarrow \bullet$

d) Fundamentos del método SLR:
- **i) Idea central:** Construir un autómata finito determinista (AFD) a partir de la gramática del lenguaje para reconocer prefijos viables, que servirá de *apoyo* para el análisis.
  - a) Observaciones:
    - (1) Tener un AFD como parte del proceso no significa que la complejidad del análisis sea comparable a la del escáner: sigue haciendo falta una pila y, por lo tanto, el proceso es el propio de un autómata a pila.
    - (2) La función de transición entre estados se codifica en la **tabla de ACCIÓN / *goto***:
      - (a) Las transiciones con terminales, en la **subtabla de ACCIÓN**;
      - (b) Las transiciones con no terminales, en la **subtabla de *goto***.
- ii) Para ello, agrupa *ítems LR*(0) en conjuntos afines, que son los que dan lugar finalmente a los estados del autómata.
  - a) El sistema de agrupación se basa en el método de construcción de un *AFD mínimo* a partir del AFND que resulta de considerar los distintos *ítems LR*(0) de la gramática.
  - b) El conjunto de estados de ese *AFD mínimo* determina la colección canónica de *ítems LR*(0) de la gramática (punto siguiente).

2. Construcción de COLECCIONES CANÓNICAS de *ítems LR*(0).

a) Gramática extendida (G_E):
Dada la gramática $G=(N,T,S,P)$ su **gramática extendida** (*augmented grammar*) se define como:

$$G_E=(N \cup \{S'\}, T, S', P \cup \{S' \rightarrow S\})$$

- **Observación:** Se acepta la cadena de entrada **sólo** cuando el analizador va a reducir con $S' \rightarrow S$

b) Operación *Cierre(I)*:

Sea *I* un conjunto de *ítems* obtenidos a partir de una cierta gramática G. Construimos *Cierre(I)* como:

1.-*Cierre(I)* ← ***I;***
2.-**SI** *([A → α • Bβ] ∈ Cierre(I)) ∧ ((B → γ) ∈ P)* ⇒
SI *([B → • γ] ∉ Cierre(I))* ⇒
Cierre(I) ← *Cierre(I)* ∪ *{[B → • γ]};*
3.-Aplicar el paso 2 hasta que no sea posible añadir nada más a *Cierre(I).*

i) Observación: Como siempre en análisis ascendente, se busca primero cómo reducir *B* antes de continuar buscando cómo reducir *A*.

c) Operación *goto(I,X)*:

Sean un conjunto de *ítems*, I, y un símbolo de la gramática, *X ∈ (N ∪ T)*. El conjunto *goto(I,X)* se define como sigue:

goto(I,X) = Cierre({[A → αX • β] | [A → α • Xβ] ∈ I })

i) Observación: Es la función de transición entre estados del AFD; se transita tras haber reconocido X (∈ N∪T)

d) Algoritmo de construcción de colecciones canónicas de ítems *LR(0)*:

ENTRADA: G_E , gramática extendida.
SALIDA: *C*, colección canónica de *ítems LR(0)* para G_E .
PROCESO:

1. *C* ← *{Cierre({[S' → • S]})};*
2. **REPETIR** (hasta que no se pueda añadir nada a C):
   a) ∀ *I ∈ C* **:**
      i) ∀ *X ∈ (N ∪ T)* **:**
         a) **SI** ((*goto(I,X)* ≠ ∅) ⇒
            (1) **SI** ((*goto(I,X)* ∉ *C*) ⇒ *C* ← *C* ∪ *goto(I,X);*

- **Ejemplo de aplicación:**

Construimos la colección canónica de *ítems LR*(0) para la gramática de expresiones aritméticas simples:

G = (N, T, E, P)
- *N = {E, T, F}*
- *T = {+, *, (,), id}*
- *P = {* *E → E+T,*
  E → T,
  *T → T*F,*
  T → F,
  F → (E),
  F → id }

1.- Determinación de la gramática extendida, G_E*:*

G_E **=(N',T,E',P'):**

- *N' = {E', E, T, F}*
- *T = {+, *, (,), id}*
- *P'={* *E' → E,* (0)
  E → E+T, (1)
  E → T, (2)
  *T → T*F,* (3)
  T → F, (4)
  F → (E), (5)
  F → id (6) *}*

2.- Determinación de la colección canónica de ítems LR(0) propiamente dicha (C):

- *C ← { I_0 }*
  - *I_0 = Cierre({[E' → • E]}) =*
    = {[E' → • E]} ∪
    ∪ {[E → • E+T], [E → • T]} ∪
    *∪ {[T → • T*F], [T → • F]} ∪*
    ∪ {[F → • (E)], [F → • id]}

  - $g_0^{E'}$ *= goto(I_0,E')= ∅;*
  - g_0^{E} *= goto(I_0,E)= {[E' → E•], [E → E• +T]} = I_1*
  - g_0^{T} *= goto(I_0,T)= {[E → T•], [T → T• *F]} = I_2*
  - g_0^{F} *= goto(I_0,F)= {[T → F•]} = I_3*
  - g_0^{+} *= goto(I_0,+)= ∅;*
  - g_0^{*} *= goto(I_0,*)= ∅;*
  - $g_0^{(}$ *= goto(I_0,()= {[F → (•E)]} ∪*
    ∪ {[E → • E+T], [E → • T]} ∪
    *∪ {[T → • T*F], [T → • F]} ∪*
    ∪ {[F → • (E)], [F → • id]} = I_4
  - $g_0^{)}$ *= goto(I_0,))= ∅;*
  - g_0^{id} *= goto(I_0,**id**)= {[F → **id**•]} = I_5*

- *C ← { I_0 ,I_1 ,I_2 ,I_3 ,I_4 ,I_5 }*
  - $g_1^{E'} = g_1^{E} = g_1^{T} = g_1^{F} = g_1^{*} = g_1^{(} = g_1^{)} = g_1^{id} = \emptyset$;
  - g_1^{+} *= goto(I_1,+) = {[E → E+ •T]} ∪ {[T → • T*F], [T → • F]} ∪ {[F → • (E)], [F → • id]} = I_6*

- *C ← { I_0 ,I_1 ,I_2 ,I_3 ,I_4 ,I_5 ,I_6 }*
  - $g_2^{E'} = g_2^{E} = g_2^{T} = g_2^{F} = g_2^{+} = g_2^{(} = g_2^{)} = g_2^{id} = \emptyset$;
  - g_2^{*} *= goto(I_2,*)= {[T → T* •F]} ∪ {[F → • (E)], [F → • id]} = I_7*

- *C ← { I_0 ,I_1 ,I_2 ,I_3 ,I_4 ,I_5 ,I_6 ,I_7 }*
  - $g_3^{E'} = g_3^{E} = g_3^{T} = g_3^{F} = g_3^{+} = g_3^{*} = g_3^{(} = g_3^{)} = g_3^{id} = \emptyset$;

- *C = { I_0 ,I_1 ,I_2 ,I_3 ,I_4 ,I_5 ,I_6 ,I_7 }* (SIN CAMBIOS)
  - $g_4^{E'} = \emptyset$;
  - g_4^{E} *= goto(I_4,E) = {[F→ (E•)], [E→ E •+T]} = I_8*
  - g_4^{T} *= goto(I_4,T) = {[E → T•], [T → T• *F]} = I_2*
  - g_4^{F} *= goto(I_4,F) = {[T → F•]} = I_3*
  - $g_4^{+} = g_4^{*} = \emptyset$;
  - $g_4^{(}$ *= goto(I_4,() = {[F → (•E)]} ∪ {[E → • E+T], [E → • T]} ∪ {[T → • T*F], [T → • F]} ∪*
    ∪ {[F → • (E)], [F → • id]} = I_4
  - $g_4^{)} = \emptyset$;
  - g_4^{id} *= goto(I_4,**id**)= {[F → **id**•]} = I_5*

- $C \leftarrow \{ I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8 \}$
  - $\forall X \in (N \cup T)\ goto(I_5, X) = \varnothing$;

- $C = \{ I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8 \}$ (SIN CAMBIOS)
  - $g_6^{E'} = g_6^{E} = \varnothing$;
  - $g_6^{T} = goto(I_6, T) = \{[E \rightarrow E + T\bullet],\ [T \rightarrow T\bullet * F]\} = I_9$
  - $g_6^{F} = goto(I_6, F) = \{[T \rightarrow F\bullet]\} = I_3$
  - $g_6^{+} = g_6^{*} = \varnothing$;
  - $g_6^{(} = goto(I_6, () = \{[F \rightarrow (\bullet E)]\} \cup \{[E \rightarrow \bullet E+T],\ [E \rightarrow \bullet T]\} \cup \{[T \rightarrow \bullet T*F],\ [T \rightarrow \bullet F]\} \cup \cup \{[F \rightarrow \bullet (E)],\ [F \rightarrow \bullet \mathbf{id}]\} = I_4$
  - $g_6^{)} = \varnothing$;
  - $g_6^{id} = goto(I_6, \mathbf{id}) = \{[F \rightarrow \mathbf{id}\bullet]\} = I_5$

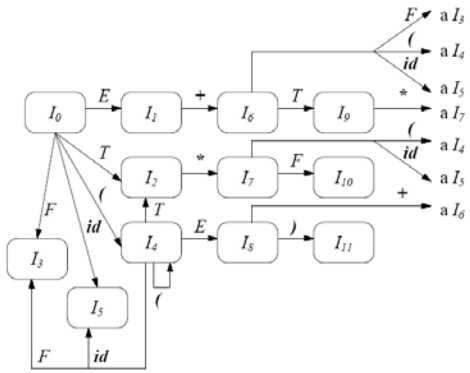
- $C \leftarrow \{ I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9 \}$
  - $g_7^{E'} = g_7^{E} = g_7^{T} = \varnothing$;
  - $g_7^{F} = goto(I_7, F) = \{[T \rightarrow T*F\bullet]\} = I_{10}$
  - $g_7^{+} = g_7^{*} = \varnothing$;
  - $g_7^{(} = goto(I_7, () = \{[F \rightarrow (\bullet E)]\} \cup \{[E \rightarrow \bullet E+T],\ [E \rightarrow \bullet T]\} \cup \{[T \rightarrow \bullet T*F],\ [T \rightarrow \bullet F]\} \cup \cup \{[F \rightarrow \bullet (E)],\ [F \rightarrow \bullet \mathbf{id}]\} = I_4$
  - $g_7^{)} = \varnothing$;
  - $g_7^{id} = goto(I_7, \mathbf{id}) = \{[F \rightarrow \mathbf{id}\bullet]\} = I_5$

- $C \leftarrow \{ I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9, I_{10} \}$
  - $g_8^{E'} = g_8^{E} = g_8^{T} = g_8^{F} = \varnothing$;
  - $g_8^{+} = goto(I_8, +) = \{[E \rightarrow E+\bullet T]\} \cup \{[T \rightarrow \bullet T*F],\ [T \rightarrow \bullet F]\} \cup \{[F \rightarrow \bullet (E)],\ [F \rightarrow \bullet \mathbf{id}]\} = I_6$
  - $g_8^{*} = g_8^{(} = \varnothing$;
  - $g_8^{)} = goto(I_8,)) = \{[F \rightarrow (E)\ \bullet]\} = I_{11}$
  - $g_8^{id} = \varnothing$;

- $C \leftarrow \{ I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9, I_{10}, I_{11} \}$
  - $g_9^{E'} = g_9^{E} = g_9^{T} = g_9^{F} = g_9^{+} = g_9^{(} = g_9^{)} = g_9^{id} = \varnothing$;
  - $g_9^{*} = goto(I_9, *) = I_7$
- $C = \{ I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9, I_{10}, I_{11} \}$ (S. C.)
  - $\forall X \in (N \cup T)\ goto(I_{10}, X) = \varnothing$;

- $C = \{ I_0, I_1, I_2, I_3, I_4, I_5, I_6, I_7, I_8, I_9, I_{10}, I_{11} \}$ (S. C.)
  - $\forall X \in (N \cup T)\ goto(I_{11}, X) = \varnothing$;

3.- *Construcción (en paralelo) del diagrama del AFD:*
Es de gran ayuda para completar las casillas con acciones de DESPLAZAMIENTO en la **subtabla de ACCIÓN** y de las casillas de la **subtabla de *goto***.



3. Algoritmo de construcción de tablas SLR(1):

ENTRADA: G_E, gramática extendida.
SALIDA: Tabla de ACCIÓN y *goto* asociada a G_E para el análisis *SLR*(1).
PROCESO:

a) Se construye la colección canónica de *ítems LR*(0) para G_E, $C = \{ I_0, \ldots, I_n \}$.
b) A partir de cada conjunto de *ítems* I_i se construye un estado *i*.
c) Las **acciones de análisis de cada estado *i*** se determinan como sigue:
   i) **SI** $([A \rightarrow \alpha\bullet\mathbf{a}\beta] \in I_i) \wedge (goto(I_i, \mathbf{a}) = I_j) \wedge (\mathbf{a} \in T)$
      $\Rightarrow$ Acción[i,a] $\leftarrow$ DESPLAZAR *j* ;
   ii) **SI** $([A \rightarrow \alpha\bullet] \in I_i)$
      $\Rightarrow \forall \mathbf{a} \in$ SIGUIENTE(A), A≠S':
         Acción[i,**a**] $\leftarrow$ REDUCIR A$\rightarrow\alpha$;
   iii) **SI** $([S' \rightarrow S\bullet] \in I_i) \Rightarrow$ Acción[i,$] $\leftarrow$ ACEPTAR;
   iv) Si se plantea algún conflicto en la generación de las tablas (es decir, si se intenta situar más de una acción en una misma casilla) $\Rightarrow$ G_E no es *SLR(1)*.
d) Las **transiciones *goto* de cada estado *i*** se construyen, para cada A∈N, según la regla:
   SI $(goto(I_i, A) = I_j) \Rightarrow goto[i,A] \leftarrow j$;
e) Todas **las casillas en blanco** de ambas subtablas **son casos de error**.
f) El **estado inicial del analizador** es el que se construye a partir del conjunto de *ítems* que contiene $[S' \rightarrow \bullet S]$.

EJEMPLO: Cálculo de las tablas de análisis *SLR*(1) para la gramática anterior.

- SIGUIENTE(E) = {+,), $}
  - SIGUIENTE(E') = {$}
- SIGUIENTE(T) = {*} ∪ SIGUIENTE(E)
- SIGUIENTE(F) = SIGUIENTE(T)

	id	+	*	(	)	$	E	T	F
0	DESPL. 5			DESPL. 4			1	2	3
1		DESPL. 6				ACEPTAR			
2		REDUCE 2	DESPL. 7		REDUCE. 2	REDUCE. 2			
3		REDUCE. 4	REDUCE. 4		REDUCE. 4	REDUCE. 4			
4	DESPL. 5			DESPL. 4			8	2	3
5		REDUCE. 6	REDUCE. 6		REDUCE. 6	REDUCE. 6			
6	DESPL. 5			DESPL. 4				9	3
7	DESPL. 5			DESPL. 4					10
8		DESPL. 6			DESPL. 11				
9		REDUCE. 1	DESPL. 7		REDUCE. 1	REDUCE. 1			
10		REDUCE. 3	REDUCE. 3		REDUCE. 3	REDUCE. 3			
11		REDUCE. 5	REDUCE. 5		REDUCE. 5	REDUCE. 5			

Algoritmo de Análisis Sintáctico LR:

1. pila.Push (s_0); *Scan* (*token*);
2. REPETIR SIEMPRE:
   a) $s_m \leftarrow$ pila.Cima();
   b) Consulta *Acción*[s_m, *token*] **:**
      i) **SI** *Acción*[s_m, *token*] = DESPLAZAR s **:**
         /* s será el siguiente estado del autómata */
         a) pila.PUSH(*token*); pila.PUSH(s); *Scan* (*token*);
      ii) **SI** *Acción*[s_m, *token*] = REDUCE $A \rightarrow \beta$ **:**
         a) EXTRAER 2*|β| símbolos de la PILA.
         /* |β| ≡ *número de símbolos de* β (longitud de β) */
         b) $s \leftarrow goto[s_{m-|\beta|}, A]$;
         /* $s_{m-|\beta|}$ ≡ *nueva cima de la PILA* */
         c) PUSH(A); PUSH(s);
         /* *No lee un nuevo token* */
      iii) **SI** *Acción*[s_m, *token*] = ACEPTAR
         ⇒ FIN(ÉXITO).
      iv) **SI** *Acción*[s_m, *token*] = ∅
         ⇒ GESTOR ERRORES.

Traza de ejecución del algoritmo para la entrada $id_1 + id_2 * id_3 \$$

ITERACIÓN	PILA	ENTRADA	ACCIÓN
Inicialización + 1	0	id_1 + id_2 * id_3 $	Desplaza (id_1, 5)
2	0 id_1 5	+ id_2 * id_3 $	Reduce (F → id)
3	0 F 3	+ id_2 * id_3 $	Reduce (T → F)
4	0 T 2	+ id_2 * id_3 $	Reduce (E → T)
5	0 E 1	+ id_2 * id_3 $	Desplaza (+, 6)
6	0 E 1 + 6	id_2 * id_3 $	Desplaza (id_2, 5)
7	0 E 1 + 6 id_2 5	* id_3 $	Reduce (F → id)
8	0 E 1 + 6 F 3	* id_3 $	Reduce (T → F)
9	0 E 1 + 6 T 9	* id_3 $	Desplaza (*, 7)
10	0 E 1 + 6 T 9 * 7	id_3 $	Desplaza (id_3, 5)
11	0 E 1 + 6 T 9 * 7 id_3 5	$	Reduce (F → id)
12	0 E 1 + 6 T 9 * 7 F 10	$	Reduce (T → T*F)
13	0 E 1 + 6 T 9	$	Reduce (E → E+T)
14	0 E 1	$	ACEPTAR