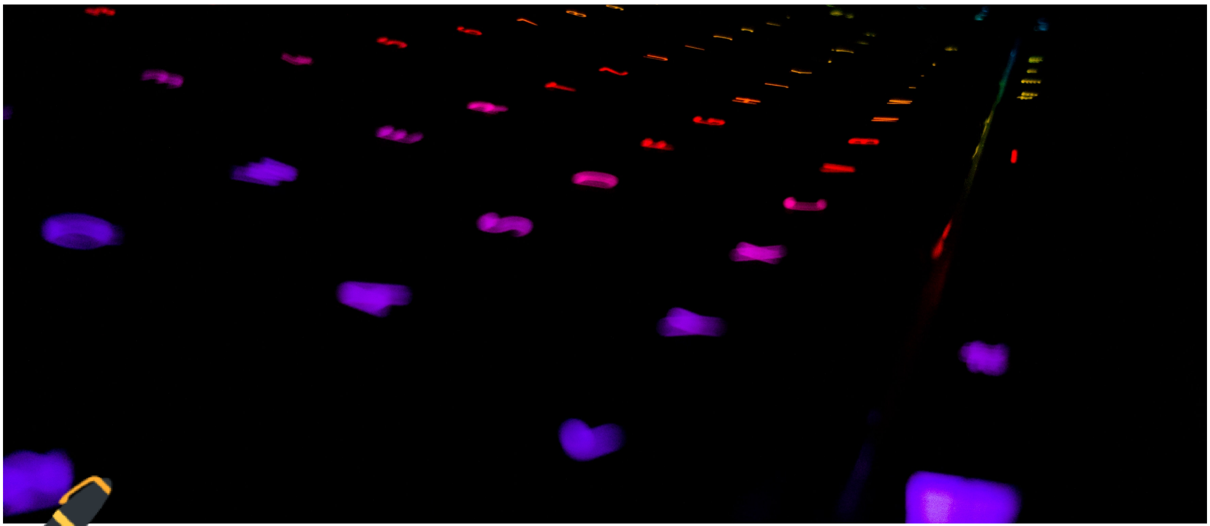


Análisis sintáctico

@Oct 27, 2020 8:45 AM-10:30 AM

Introducción

📖 **Sintaxis**: estructura del lenguaje.

⚠️ En este tema se estudiará si una cadena de tokens pertenece o no a una determinada gramática de contexto libre que genera un lenguaje.

Para ello será necesario un Autómata con Pila, como se vió en el Tema 1.

▾ $G = (N, T, P, S)$

Gramática de contexto libre.

- N : Símbolos no terminales (mayúsculas).
- T : Símbolos terminales (minúsculas y ε).
- P : Reglas de la forma $N \rightarrow (N \cup T)$.
- S : Axioma (símbolo de N inicial). *Suele ser el de la primera regla de P.*

Ejemplo

Sea la gramática G definida abajo, ¿ab $\in L(G)$?

$G = (\{A, B\}, \{a, b\}, \{A \rightarrow aBb, A \rightarrow Bab, B \rightarrow aB, B \rightarrow \varepsilon\}, A)$

$A \rightarrow aBb$

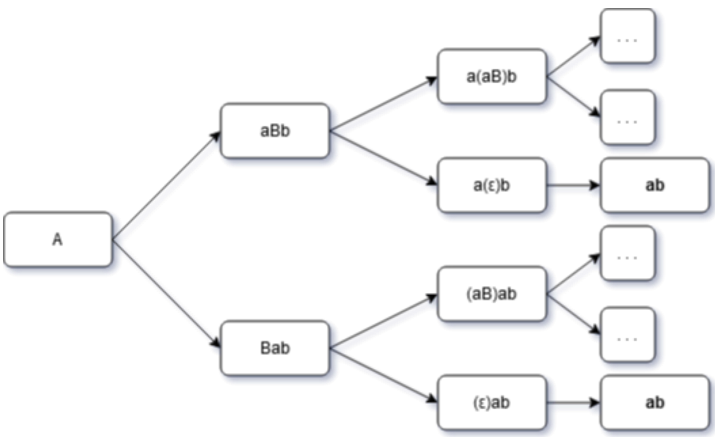
$A \rightarrow Bab$

$B \rightarrow aB$

$B \rightarrow \varepsilon$

Este proceso es de $\mathrm{O}(k^n)$, donde:

- k es el núm. medio de reglas.
- n la longitud de la cadena.



Como una rama (mínimo) de G lleva a ab, entonces $ab \in L(G)$

Como se ha visto, el coste es bastante alto, por lo que se quiere predecir qué reglas aplicar para conseguir una determinada cadena con menor coste, en lugar de expandir todas las ramas de G.

Para ello, se usará el **algoritmo LL(1)**, que se verá más adelante.

Gramática ampliada

📖 Sea una gramática $G = (N, T, P, S)$, una gramática ampliada no es más que una gramática $G' = (N', T' \cup \{\$\}, P', S')$, donde $\$$ indica el final de una cadena.

- $N' = N \cup \{S'\}$
- $T' = T \cup \{\$\}, \quad \$ \notin T$
- $P' = P \cup \{S' \to S\$\}$
- S' : Nuevo símbolo inicial.

Ejemplo

Gramática original G

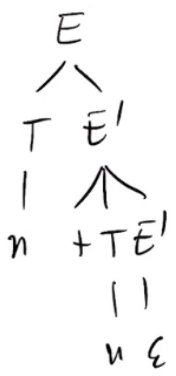
$E \to TE'$

$E' \to +TE'$

$E' \to \varepsilon$

$T \to (E)$

$T \to n$



Gramática ampliada G'

$S \to E\$$

$E \to TE'$

$E' \to +TE'$

$E' \to \varepsilon$

$T \to (E)$

$T \to n$

⚠️ Esto cobra más importancia en el apartado del **símbolo director**.

Análisis LL(1)

Funciones

Cabecera - CAB(X)

📖 $\mathrm{CAB}: \ (N \cup T)^* \longrightarrow \mathcal{P}(T) \cup \{\varepsilon\}$

📖 La cabecera de algo que empieza por terminal, es ese mismo terminal.

$$\mathrm{CAB}(\alpha) = \{\alpha\}$$
$$\mathrm{CAB}(\alpha A) = \{\alpha\}$$

$\alpha \in T, \ A \in N^+$

📖 La cabecera de ε es ε si está aislado; si no, es la de lo que le sigue.

$$\mathrm{CAB}(\varepsilon) = \{\varepsilon\}$$
$$\mathrm{CAB}(\varepsilon x) = \mathrm{CAB}(x)$$

$x \in (N \cup T)^+$

Desarrollar:

Sea la regla $X \to \alpha A \beta$:

- X es el antecedente de la regla.
- $\alpha A \beta$ es el consecuente de la regla.

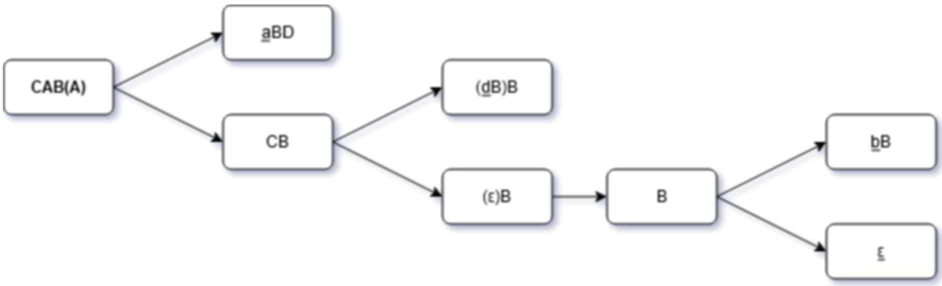
1. Desarrollar los consecuentes cuyo antecedente sea el primer término de la cabecera.
2. Si algún consecuente empieza por un símbolo terminal, dicho término se añade a la solución si no está incluido.
3. Si se han añadido **todos los símbolos terminales**, se encontró la cabecera.
   Si **todos los desarrollos** ya tienen un símbolo terminal, se encontró la cabecera.

Ejemplos

Sea la gramática G especificada por estas reglas P descritas, calcular $\mathrm{CAB}(A)$ y $\mathrm{CAB}(ABDC)$.

$S \to A\$$
$A \to aBD$
$A \to CB$
$B \to bB$
$B \to \varepsilon$
$C \to dBc$
$C \to \varepsilon$
$D \to aB$
$D \to d$

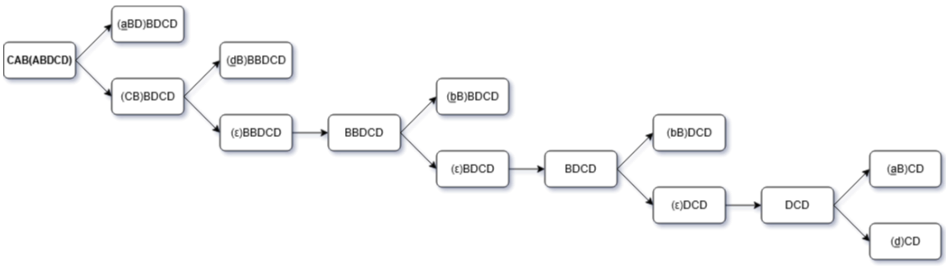
$$\mathrm{CAB}(A) = \{a, b, d, \varepsilon\}$$



🧠 Cuando la cabecera tiene más de un no-terminal, deben arrastrarse en el desarrollo.

$S \to A\$$
$A \to aBD$
$A \to CB$
$B \to bB$
$B \to \varepsilon$
$C \to dBc$
$C \to \varepsilon$
$D \to aB$
$D \to d$

$$\mathrm{CAB}(ABDCD) = \{a, b, d\}$$



Siguiente - SIG(X)

📖 $\mathrm{SIG}: \; N \longrightarrow \mathcal{P}(T) \cup \{\$\}$

🧠 $\varepsilon \notin \mathrm{SIG}(x)$

📖 Sea un conjunto de reglas $\{X_i \rightarrow \alpha_i A_i \beta_i,\ \dots,\ X_n \rightarrow \alpha_n A_n \beta_n\}$, siendo n el número de reglas de P que tienen A en su consecuente, entonces:

$\text{SIG}(A) = (\text{SIG}(A_1) \cup \cdots \cup \text{SIG}(A_n)) - \{\varepsilon\}$, donde:

$$\text{SIG}(A_i) = \begin{cases} \text{CAB}(x_i) & : x_i \neq \varepsilon \\ \text{SIG}(X_i) & : x_i = \varepsilon \end{cases}, \quad x_i, \alpha_i, \beta_i \in (N \cup T), \quad 1 \leq i \leq n \in \mathbb{N}$$

🧠 Sea $\text{SIG}(X)$ y una regla con varios A en su consecuente, todos deben tomarse en cuenta.

Por ejemplo: $X \rightarrow aBbBCB$ entonces $\text{SIG}(B) = \text{CAB}(b) \cup \text{CAB}(C) \cup \text{CAB}(\varepsilon)$.

Ejemplo

Sea la gramática G especificada por estas reglas P descritas, calcular $\text{SIG}(B)$.

$S \rightarrow A\$$

$A \rightarrow aBD$

$A \rightarrow CB$

$B \rightarrow bB$

$B \rightarrow \varepsilon$

$C \rightarrow dBc$

$C \rightarrow \varepsilon$

$D \rightarrow aB$

$D \rightarrow d$

$\text{SIG}(B) = \{a, d\} \cup \{\$\} \cup \{c\} = \{a, d, c, \$\}$

$A \rightarrow aBD : \text{CAB}(D) = \{a, d\}$

$A \rightarrow CB : \text{CAB}(\varepsilon)$

$\text{SIG}(A) = \{\$\}$

$S \rightarrow A\$: \text{CAB}(\$) = \{\$\}$

$B \rightarrow bB : \text{CAB}(\varepsilon)$

$\text{SIG}(B) = \dots$ ← **Bucle**: no aporta información, se ignora.

$C \rightarrow dBc : \text{CAB}(c) = \{c\}$

$D \rightarrow aB : \text{CAB}(\varepsilon)$

$\text{SIG}(D) = \{\$\}$

$A \rightarrow aBD : \text{CAB}(\varepsilon)$

$\text{SIG}(A) = \{\$\}$

He ignorado los ε en vez de añadirlos y eliminarlos al final.

Símbolo Director - SD(X → x)

$$\mathrm{SD} : P \longrightarrow \mathcal{P}(T \cup \{\$\})$$

Símbolos de T que pueden aparecer a la izquierda si se elige usar una regla de P.

$$\mathrm{SD}(X \to x) = \begin{cases} \mathrm{CAB}(x) & : \ \varepsilon \notin \mathrm{CAB}(x) \\ (\mathrm{CAB}(x) - \{\varepsilon\}) \cup \mathrm{SIG}(X) & : \ \varepsilon \in \mathrm{CAB}(x) \end{cases}$$

$$\mathrm{SD}(A \to \alpha) = \{\alpha \in (T \cup \{\$\}) \mid S' \Rightarrow \alpha\beta\}, \quad \alpha\beta \in (N \cup T)^*$$

$$\mathrm{SD}(X \to \varepsilon) = (\mathrm{CAB}(\varepsilon) - \{\varepsilon\}) \cup \mathrm{SIG}(X) = \emptyset \cup \mathrm{SIG}(X) = \mathrm{SIG}(X)$$

Ejemplo

$G = (\{E, E', T\}, \{n, (,), +\}, \{E \to TE', E' \to +TE', E' \to \varepsilon, T \to (E), T \to n\}, E)$

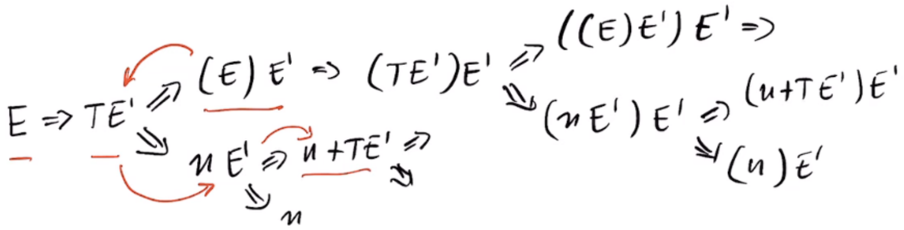
$E \to TE'$

$E' \to +TE'$

$E' \to \varepsilon$

$T \to (E)$

$T \to n$



La línea roja indica backtracking para producir la cadena `n+n`.

Sea G la gramática dada, calcular los símbolos directores de $E \to TE'$, $E' \to +TE'$ y $E' \to \varepsilon$.

$S \to E\$$

$E \to TE'$

$E' \to +TE'$

$E' \to \varepsilon$

$T \to (E)$

$T \to n$

$T \to \varepsilon$

$$\text{SD}(E \to TE') = (\text{CAB}(TE') - \{\varepsilon\}) \cup \text{SIG}(E) = \{(, n, +, \$,)\} = T$$

$$\text{CAB}(TE') = \{(, n, +, \varepsilon\}, \quad \varepsilon \in \text{SD}(E \to +TE')$$

$$\text{SIG}(E) = \{\$\} \cup \{)\} = \{), \$\}$$

$$S \to E\$: \quad \text{CAB}(\$) = \{\$\}$$

$$T \to (E) : \quad \text{CAB}(\,)\,) = \{)\}$$

$$\text{SD}(E' \to +TE') = \text{CAB}(+TE') = \{+\}$$

$$\text{CAB}(+TE') = \{+\}, \quad \varepsilon \notin \text{CAB}(+TE')$$

$$\text{SD}(E' \to \varepsilon) = \text{SIG}(E') = T$$

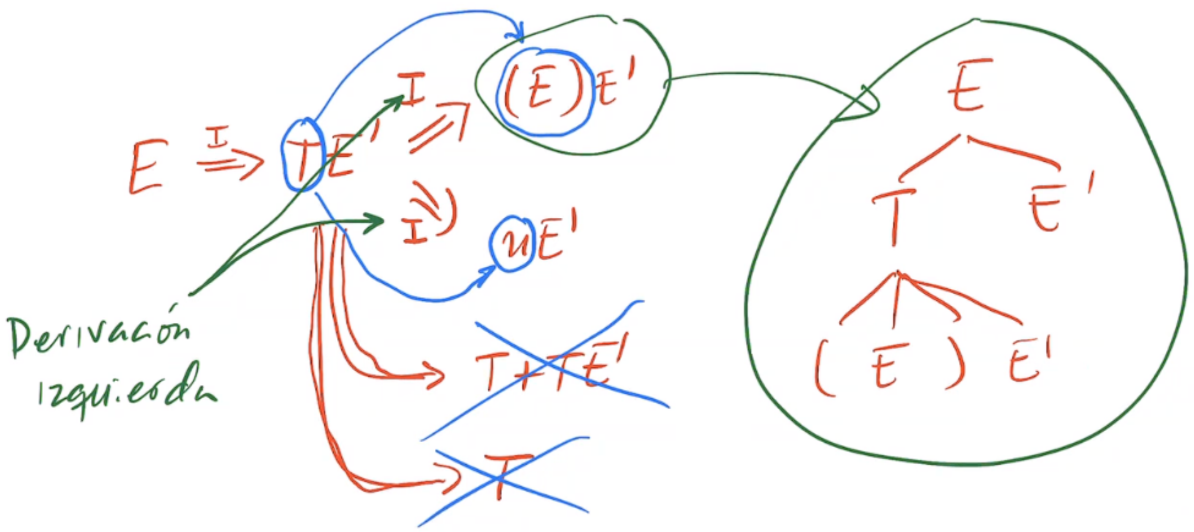
$$\text{CAB}(\varepsilon) = \{\varepsilon\}, \quad \varepsilon \in \text{CAB}(\varepsilon)$$

$$\text{SIG}(E') = (\{(, n, +, \varepsilon\} \cup \{\$,)\} \cup \{+\}) - \{\varepsilon\} = \{(, n, +, \$,)\} = T$$

$$E \to TE' : \quad \text{CAB}(TE') = \{(, n, +, \varepsilon\}, \quad \varepsilon \in \text{CAB}(TE')$$

$\text{SIG}(E) = \{), \$\}$ ← Ya se calculó antes

$$E' \to +TE' : \quad \text{CAB}(+TE') = \{+\}, \quad \varepsilon \notin \text{CAB}(+TE')$$



En un sistema de producción se usan las partes izquierdas, lo que genera **derivaciones izquierdas**.

@Oct 29, 2020 8:45 AM-10:30 AM

Condición LL(1)

📖 Una **gramática** es LL(1) si:
$\forall A \in N,\ \forall (A \to \alpha, A \to \beta) \in P \implies \mathrm{SD}(A \to \alpha) \cap \mathrm{SD}(A \to \beta) = \varnothing$

🧠 **No todas las gramáticas cumplen la condición LL(1)**, para resolver ese problema puede emplearse una **gramática equivalente**.

⚠️ Por ejemplo, las gramáticas recursivas por la izquierda, que tienen reglas del tipo $X \to A\alpha$ donde: $A \in N,\ \alpha \in T$.

📖 Un **lenguaje** es LL(1) si: $\quad \exists G \mid L(G), \quad G \text{ cumple la condición LL}(1)$

Gramáticas equivalentes

📖 Se dice que 2 gramáticas G_1 y G_2 son equivalentes si: $L(G_1) = L(G_2), \quad G_1 \neq G_2$.

Eliminación de recursividad IZQ

🧠 Este procedimiento **no garantiza que la gramática resultante sea LL(1)**, sino que se usa frecuentemente para el caso contrario: encontrar una gramática que no es LL(1) garantizada.

Ejemplo

La gramática $G_1 = \{\{A\}, \{a_i, b_i\}, \{A \to Aa_1|Aa_2|...), A \to b_1|b_2|...\}, A\}, \quad i \in \mathbb{N}$, genera el lenguaje $L(G_1) = ((b_1|b_2|...)(a_1|a_2|...))^*$.

En este caso, G_1 presenta recursividad izquierda, por lo que se busca una gramática equivalente G_2 que genere el mismo lenguaje, pero que no tenga ese problema.

Una posible solución sería $G_2 = \{\{A, B\}, \{a, b\}, \{A \to bB, B \to aA, B \to \varepsilon\}, A\}$, que genera el lenguaje $L(G_2) = L(G_1) = ((b_1|b_2|...)(a_1|a_2|...))*$.

Factorización

Sea $\alpha \in (N \cup T)^+ - \{\varepsilon\}$.

Gramática G	Transición	Gramática G'	Equivalencia
$A \to \alpha\beta$	$A \to \alpha \quad (\beta)$	$A \to \alpha B$	$L(G) = L(G')$
$A \to \alpha\gamma$	$A \to \alpha \quad (\gamma)$	$B \to \beta$	
		$B \to \gamma$	

Tabla LL(1)

📖 $\text{TablaLL}(1) : (N \cup T \cup \{\$\}) \longrightarrow (N \cup T)^*$

📖 $\text{TablaLL}(1)(A,\beta) = \begin{cases} \alpha & : \beta \in \text{SD}(A \to \alpha) \\ \emptyset & : \beta \notin \text{SD}(A \to \alpha) \end{cases}$

⚠️ Básicamente, consiste en definir **todos los símbolos directores de P**.

🧠 Si una fila de la TablaLL(1) contiene más de una casilla repetida, la gramática no cumple la condición.

🧠 Si 2 reglas con el mismo precedente tienen el mismo terminal en el consecuente, no se cumple la condición LL(1).

Ejemplo

Sea G la gramática dada, desarrollar la Tabla LL(1).

$S \to E\$$	$\text{SD}(S \to E\$) = \text{CAB}(E\$) = \{(, n\}$
$E \to TE'$	$\text{SD}(E \to TE') = \text{CAB}(TE') = \{(, n\}$
$E' \to +TE'$	$\text{SD}(E' \to +TE') = \text{CAB}(+TE') = \{+\}$
$E' \to \varepsilon$	$\text{SD}(E' \to \varepsilon) = \text{SIG}(E') = \text{SIG}(E) = \{\$,)\}$
$T \to (E)$	$\text{SD}(T \to (E)) = \text{CAB}((E)) = \{(\}$
$T \to n$	$\text{SD}(T \to n) = \text{CAB}(n) = \{n\}$

🧠 La tabla se crea colocando **cada elemento de T** arriba y **cada elemento de N** en el lateral; y se rellena con los **consecuentes de cada regla de P**, comprobando cuáles son los símbolos directores de cada uno.

⚠️ Por ejemplo, para la primera regla:
$\text{SD}(S \to E\$) = \{(, n\} \implies \text{TablaLL}(1)(S, () = \text{TablaLL}(1)(S, n) = E\$$

TablaLL(1)

	n	**(**	**)**	**+**	**$**
S	$E\$$	$E\$$			
E	TE'	TE'			
E'			ε	$+TE'$	ε
T	n	(E)			

Algoritmo LL(1)

Paso básico:

- Insertar el estado inicial en la pila (suele ser S).
- La entrada es el primer símbolo.

<table>
<tr><th>Pila</th><th>Lexema</th></tr>
<tr><td>[S]</td><td><code>| h |o|l|a| |m|u|n|d|o|$|</code></td></tr>
</table>

🧠 **La pila se lee hacia abajo**; es decir, se inserta «al revés».

⚠️ Yo hago la pila en horizontal para mis apuntes; por tanto, mi «izquierda» en horizontal es «abajo» en vertical.

Desarrollo:

- Sea $x \in (N \cup T \cup \{\$\})^*$ el símbolo de la cima de la pila.
- Sea $a \in T \cup \{\$\}$ el símbolo de la entrada.
- Sea un lexema de la forma $T^*\$$.

1. Paso básico (inicializar la pila y el lexema).
2. Bucle:

1. Si $x \in N$:
    - $\text{TablaLL}(1)(\text{x, a}) \neq \emptyset$: Sustituir x en la pila por a.
    - $\text{TablaLL}(1)(\text{x, a}) = \emptyset$: Error.
2. Si $x \in T$:
    - $x = a$: Avanzar en la lectura y eliminar x de la pila.
    - $x \neq a$: Error.
3. Salir del bucle si (**alguna**):
    - Hubo un error.
    - La pila está vacía.
    - La entrada está vacía.

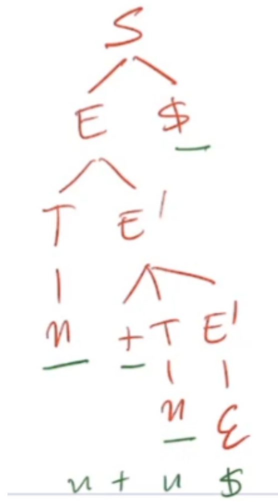
3. Si la pila o la entrada se vacía, entonces $c \in L$.
   Si hubo algún error, entonces $c \notin L$.

Ejemplo

Sea la cadena `n+n$` y la gramática dada, ¿$\text{n+n\$} \in L(G)$?

<table>
<tr><th>Gramática</th><th>Fase</th><th>Pila</th><th>Cadena</th></tr>
<tr><td>$S \to E\$$</td><td>1</td><td>[S]</td><td>• | n |+|n|$|</td></tr>
<tr><td>$E \to TE'$</td><td>2</td><td>[$, E]</td><td>• | n |+|n|$|</td></tr>
<tr><td>$E' \to +TE'$</td><td>3</td><td>[$, E' , T]</td><td>• | n |+|n|$|</td></tr>
<tr><td>$E' \to \varepsilon$</td><td>4</td><td>[$, E' , n]</td><td>• | n |+|n|$|</td></tr>
<tr><td>$T \to (E)$</td><td>5</td><td>[$, E']</td><td>• |n| + |n|$|</td></tr>
<tr><td>$T \to n$</td><td>6</td><td>[$, E' , T , +]</td><td>• |n| + |n|$|</td></tr>
<tr><td></td><td>7</td><td>[$, E' , T]</td><td>• |n|+| n |$|</td></tr>
<tr><td></td><td>8</td><td>[$, E' , n]</td><td>• |n|+| n |$|</td></tr>
<tr><td></td><td>9</td><td>[$, E']</td><td>• |n|+|n| $ |</td></tr>
<tr><td>⚠ TablaLL(1) necesaria.</td><td>10</td><td>[$ _] → [$]</td><td>• |n|+|n| $ |</td></tr>
<tr><td></td><td>11</td><td>[ ]</td><td>• |n|+|n|$|</td></tr>
</table>

1. Inicio: se añade S a la pila y se empieza leyendo desde n
2. $\text{TablaLL}(1)(S, n) = E\$$ → Sustituir S por $E\$$
3. $\text{TablaLL}(1)(E, n) = TE'$ → Sustituir E por TE'
4. $\text{TablaLL}(1)(T, n) = n$ → Sustituir T por n
5. Como $n \in T$ → Se avanza en la lectura y se elimina n
6. $\text{TablaLL}(1)(E', +) = +TE'$ → Sustituir E' por $+TE'$
7. Como $+ \in T$ → Se avanza en la lectura y se elimina $+$
8. $\text{TablaLL}(1)(T, n) = n$ → Sustituir T por n
9. Como $n \in T$ → Se avanza en la lectura y se elimina n
10. $\text{TablaLL}(1)(E', \$) = \varepsilon$ → Sustituir E' por ε



🧠 Sustituir un término por ε equivale a eliminarlo

11. Como $\varepsilon \in T$ → Se avanza en la lectura y se elimina $

Como la pila está vacía, la entrada llegó al final y no han habido errores, $\text{n+n\$} \in L(G)$ *usando la secuencia indicada.*

@Nov 10, 2020 8:45 AM-10:30 AM

Análisis SLR(1)

Ítem LR(0)

📖 Sea una gramática de contexto libre $G = (N, T, P, S)$.

Un **ítem LR(0)** es una terna (A, α, β) donde:
$A \in N,\ \alpha, \beta \in (N \cup T)^*,\ (A \to \alpha\beta) \in P.$

🧠 Otra representación de un

🧠 La **cantidad posible** de ítems

ítem LR(0) es $[A \to \alpha \cdot \beta]$, y **suele usarse más**.

LR(0) de cualquier gramática es **finita**.

Ejemplo

Sea la gramática G definida por las siguientes reglas de P, **¿cuántos ítems LR(0) hay?**

Algunos de los ítems LR(0) son:

$S \to E\$$	$(E, \varepsilon, E\$)$	$(E, E, \$)$	$(E, E\$, \varepsilon)$	
$E \to E + T$	$(E, \varepsilon, E + T)$	$(E, E+, T)$	$(E, E + T, \varepsilon)$	$\dots$
$E \to T$	(E, ε, T)	(E, T, ε)		
$T \to (E)$	$\vdots$	$\vdots$	$\vdots$	
$T \to n$	$\vdots$	$\vdots$	$\vdots$	

Expresando las reglas como $[A \to \alpha \cdot \beta]$ puede resultar más fácil, ya que moviendo el $\cdot$ entre los símbolos se obtienen todas las combinaciones posibles.

Bastaría con contar cada posición posible del $\cdot$ en cada regla:

Algunos ítems LR(0) en la forma $[A \to \alpha \cdot \beta]$ son:

$S \to \cdot E \cdot \$ \cdot$	3	$S \to \varepsilon \cdot E\$$	$S \to E \cdot \$$	$S \to E\$ \cdot \varepsilon$
$E \to \cdot E \cdot + \cdot T \cdot$	4			
$E \to \cdot T \cdot$	2			
$T \to \cdot (\cdot E \cdot) \cdot$	4			
$T \to \cdot n \cdot$	2			
	15			

Existen 15 ítems LR(0) para la gramática G.

Sea una gramática de contexto libre $G = (N, T, P, S)$.

Un **ítem LR(0) completo** es un ítem de la forma $[A \to \alpha \cdot] \equiv (A, \alpha, \varepsilon)$.

Un ítem LR(0) completo es un ítem LR(0) terminado en $\cdot$.

Operaciones con ítems LR(0) - Funciones

Se denomina $\emptyset$ al conjunto de **todos los ítems LR(0) de una gramática** G.

Sea I un conjunto de ítems LR(0) de una gramática G, tal que $I \in \emptyset$.

Para evitar futuras confusiones:
$\emptyset$: todos los ítems (estrecho).
$\varnothing$: conjunto vacío (redondo).

Cierre - CIERRE(I)

$\mathrm{CIERRE}(I) : \mathcal{P}(\emptyset) \longrightarrow \mathcal{P}(\emptyset)$

$\mathrm{CIERRE}(I) = I \cup \{[A \to \cdot\, \alpha] \in \emptyset \mid \exists [X \to \beta \cdot A\gamma] \in \mathrm{CIERRE}(I)\}$, donde:

$$\alpha, \beta, \gamma \in (N \cup T)^*, \quad A, X \in N$$

Ejemplo

Sea $I = \{E \to E + \cdot T, E \to T \cdot, T \to (\cdot E), T \to n \cdot\}$, calcular $\mathrm{CIERRE}(I)$.

$E \to E+\cdot T$

$E \to T\cdot$

$T \to (\cdot E)$

$T \to n\cdot$

1. { E → E+·T, E → T·, T → (·E), T → n· }
2. { **E → E+·T**, E → T·, T → (·E), T → n·, T → ·(E), T → ·n }
3. { **E → E+·T**, E → T· , T → (·E), T → n·, T → ·(E), T → ·n }
4. { **E → E+·T**, **E → T·**, T → (·E) , T → n· , T → ·(E), T → ·n }
5. { **E → E+·T**, **E → T·**, **T → (·E)**, T → n·, T → ·(E), T → ·n, E → ·E+T, E → ·T }
6. { **E → E+·T**, **E → T·**, **T → (·E)**, T → n· , T → ·(E), T → ·n, E → ·E+T, E → ·T }
7. { **E → E+·T**, **E → T·**, **T → (·E)**, **T → n·**, T → ·(E), T → ·n, E → ·E+T, E → ·T }
8. { **E → E+·T**, **E → T·**, **T → (·E)**, **T → n·**, **T → ·(E)**, T → ·n, E → ·E+T, E → ·T }
9. { **E → E+·T**, **E → T·**, **T → (·E)**, **T → n·**, **T → ·(E)**, **T → ·n**, E → ·E+T, E → ·T }
10. { **E → E+·T**, **E → T·**, **T → (·E)**, **T → n·**, **T → ·(E)**, **T → ·n**, **E → ·E+T**, E → ·T }
11. { **E → E+·T**, **E → T·**, **T → (·E)**, **T → n·**, **T → ·(E)**, **T → ·n**, **E → ·E+T**, **E → ·T** }

$$\text{CIERRE}(I) = \{E \to E+\cdot T, E \to T\cdot\ , T \to (\cdot E), T \to n\cdot\ , T \to\cdot (E), T \to\cdot n, E \to\cdot E+T, E \to\cdot T\}$$

1. Se lee lo siguiente al · del primer ítem, como es un no-terminal (T) que no ha salido:
2. Se expanden las reglas de dicho no-terminal, con el · al inicio de cada consecuente.
3. Se lee lo siguiente al · del segundo ítem, como es el terminal ε, esa regla ya está **cerrada**.
4. Se lee lo siguiente al · del tercer ítem, como es un no-terminal (E) que no ha salido:
5. Se expanden las reglas de dicho no-terminal, con el · al inicio de cada consecuente.
6. Se lee lo siguiente al · del cuarto ítem, como es el terminal ε, esa regla ya está **cerrada**.

7. Se lee lo siguiente al · del quinto ítem, como es el terminal (, esa regla ya está **cerrada**.
8. Se lee lo siguiente al · del sexto ítem, como es el terminal n, esa regla ya está **cerrada**.
9. Se lee lo siguiente al · del séptimo ítem: el no-terminal T ya se expandió, esa regla se **cierra**.
10. Se lee lo siguiente al · del octavo ítem: el no-terminal E ya se expandió, esa regla se **cierra**.
11. Se han analizado el cierre de todos los ítems, por lo que este conjunto es $\text{CIERRE}(I)$.

Núcleo - NÚCLEO(I)

📖 $\text{NÚCLEO}(I) : \mathcal{P}(\emptyset) \longrightarrow \mathcal{P}(\emptyset)$

🧠 $\text{NÚCLEO}(I)$ es único.

📖 Sean I, J, K conjuntos de ítems LR(0):

$$\text{CIERRE}(\text{NÚCLEO}(I)) = I, \quad \forall K \mid \text{CIERRE}(K) = I, \ |\text{NUCLEO}(I)| < |K|$$

🧠 $\text{NÚCLEO}(I)$ contiene el mínimo número de elementos de I.

Ejemplo

Sea I el conjunto definido por las reglas de abajo, calcular $\text{NÚCLEO}(I)$.

$S \rightarrow E \cdot \$$

$E \rightarrow \cdot E + T$

$E \rightarrow \cdot T$

$T \rightarrow (\cdot E)$

$T \rightarrow \cdot (E)$

Sean los conjuntos:

$J = \{S \rightarrow E \cdot \$, T \rightarrow (\cdot E)\}$

$K = \{S \rightarrow E \cdot \$, E \rightarrow \cdot E + T, T \rightarrow (\cdot E)\}$

$\text{CIERRE}(J) = \text{CIERRE}(K) = I$, *pero como* $|J| < |K| \Rightarrow \text{NÚCLEO}(I) = J$.

$T \rightarrow \cdot n$

Delta - DELTA(I, X)

📖 $\delta(I) : \mathcal{P}(\emptyset) \times (N \cup T \cup \{\$\}) \longrightarrow \mathcal{P}(\emptyset)$

📖 $\delta(I, X) = \text{CIERRE}(J), \quad J = \{[A \rightarrow \alpha X \cdot \beta] \in \emptyset \mid [A \rightarrow \alpha \cdot X \beta] \in I\}$

Ejemplo

Sea I el conjunto definido por las reglas de abajo, calcular $\delta(I, n), \delta(I, E), \delta(I, +)$.

$S \rightarrow E\$$	$E \rightarrow \cdot E + T$	$T \rightarrow (\cdot E)$
	$E \rightarrow E \cdot + T$	$T \rightarrow n \cdot$

Se buscan las reglas cuyo consecuente sea $\alpha \cdot n \beta$;
por tanto, $J = \varnothing$ porque no hay.

$\delta(I, n) = \text{CIERRE}(J) = \varnothing$

Se buscan las reglas cuyo consecuente sea $\alpha \cdot E \beta$;
por tanto, $J = \{E \rightarrow E \cdot + T, T \rightarrow (E \cdot)\}$.

$\delta(I, E) = \text{CIERRE}(J) = \{E \rightarrow E \cdot + T, T \rightarrow (E \cdot)\}$

Se buscan las reglas cuyo consecuente sea $\alpha \cdot + \beta$;
por tanto, $J = \{E \rightarrow E + \cdot T\}$.

$\delta(I, +) = \text{CIERRE}(J) = \{E \rightarrow E + \cdot T, T \rightarrow \cdot (E), T \rightarrow \cdot n\}$

Colección de ítems LR(0)

📖 Se define como un conjunto compuesto de ítems LR(0).
$\mathbb{C} = \{I_0, I_1, \ldots, I_n\}, \quad I_n \subseteq \mathcal{P}(\emptyset), \quad n \in \mathbb{N}$

📖 $I_0 \equiv \text{CIERRE}(\{[S' \to \cdot S\$]\})$: el cierre de la primera regla.

🧠 I_0 es el conjunto $\emptyset_G$, pero con el $\cdot$ al inicio de todos los consecuentes.

Desarrollo:

- Sea el conjunto $Nuevo = \{I_0\}$.
- Sea el conjunto $Viejo = \varnothing$.

1. Bucle:
   1. $Viejo = Viejo \cup Nuevo$
   2. Bucle:
      - Sea $I_k \in Nuevo$.
      - $X \in (N \cup T \cup \{\$\})$
      - $k \in [0, |Nuevo| - 1]$

      1. $I_{k+1} = \delta(I_k, X)$
      2. $Nuevo = Nuevo \cup I_{k+1}$
      3. Si $k = |Nuevo| - 1$ salir.
   3. Si $Viejo = Nuevo$ salir.
2. $\mathbb{C} = Viejo = Nuevo$

Ejemplo

Sea G la gramática definida por las reglas de abajo, calcular $\mathbb{C}$.

Gramática G	I_0

$$\begin{array}{lll}
S \to E\$ & S \to \cdot E\$ & \delta(I_0, E) = I_1 \\
E \to E + T & E \to \cdot E + T & \delta(I_0, T) = I_2 \\
E \to T & E \to \cdot T & \delta(I_0, \$) = \varnothing \\
T \to (E) & T \to \cdot (E) & \delta(I_0, +) = \varnothing \\
T \to n & T \to \cdot n & \delta(I_0, \ () = I_3 \\
 & & \delta(I_0, \ )\ ) = \varnothing \\
 & & \delta(I_0, n) = \{T \to n \cdot\}
\end{array}$$

$$\delta(I_0, E) : J_1 = \{S \to E \cdot \$, E \to E \cdot +T\}$$
$$\delta(I_0, E) = \text{CIERRE}(J_1) = I_1$$

$$\delta(I_0, T) : J_2 = \{E \to T \cdot\}$$
$$\delta(I_0, T) = \text{CIERRE}(J_2) = I_2$$

$$\delta(I_0, \ () : J_3 = \{T \to (\cdot E)\} \cup \{E \to \cdot E + T, E \to \cdot T\} \cup \{T \to \cdot (E), T \to \cdot n\}$$
$$\delta(I_0, \ () = \text{CIERRE}(J_3) = I_3$$

Habría que seguir calculando $\delta(I_1, \$)$, $\delta(I_1, +)$ y todas las funciones δ con I_n que surgieran.

⚠ Esto solo ha sido para calcular I_0.

⚠ Esta forma es muy jodidamente ineficiente, hay otra mejor.

Tabla de transiciones

📖 $\text{TablaTransiciones} : (\mathbb{C} \times N \cup T \cup \{\$\}) \longrightarrow \mathbb{C}$

📖 $\text{TablaTransiciones}(I, X) \equiv \delta(I, X)$

🧠 Representación de $\delta(I, X)$ en formato de tabla.

Ejemplo

Sea G la gramática definida por las reglas de abajo, escribir la tabla de transiciones.

$S \to E\$$

$E \to E + T$

$E \to T$

$T \to (E)$

$T \to n$

⚠️ $\mathbb{C}$ **necesario.**

⚠️ He marcado las líneas vacías para que se identifiquen mejor, no es un símbolo especial ni nada por el estilo.

	(	**n**	**)**	**+**	**$**	**E**	**T**
I_0	I_3	I_4				I_1	I_2
I_1				I_6	I_5		
I_2	-	-	-	-	-	-	-
I_3	I_3	I_4				I_7	I_2
I_4	-	-	-	-	-	-	-
I_5	-	-	-	-	-	-	-
I_6	I_3	I_4					I_8
I_7			I_9	I_6			
I_8	-	-	-	-	-	-	-
I_9	-	-	-	-	-	-	-

Tabla de acciones

📖 $\text{TablaAcciones} : (\mathbb{C} \times T \cup \{\$\}) \longrightarrow \{\text{Desplazamiento, Reducción, Aceptar, Error}\}$

📖 $$\text{TablaAcciones}(I, x) = \begin{cases} \text{Desplazamiento} & : \exists[X \to \alpha \cdot x\beta] \in I \\ \text{Reducción} & : \exists[X \to \alpha \cdot] \in I, \quad x \in SIG(X) \\ \text{Aceptar} & : \exists[S \to S\$ \cdot] \in I \\ \text{Error} & : \text{Nada de lo anterior} \end{cases}$$

Ejemplo

Sea G la gramática definida por las reglas de abajo, escribir la tabla de transiciones.

$S \to E\$$

$E \to E + T$

$E \to T$

$T \to (E)$

$T \to n$

🧠 Cada acción se representa con su inicial.

⚠️ TablaTransiciones **necesaria.**

⚠️ Orden que he seguido: D → R → A → E (casillas vacías).

	(	**n**	**)**	**+**	**$**
I_0	D	D			
I_1				D	D
I_2			R (E→T)	R (E→T)	R(E→T)
I_3	D	D			
I_4			R (T→n)	R (T→n)	R (T→n)

I_5	A	A	A	A	A
I_6	D	D			
I_7			D	D	
I_8			R (T→(E))	R (T→(E))	R (T→(E))
I_9			R (E→E+T)	R (E→E+T)	R (E→E+T)

🧠 Puede haber más de una acción en una misma fila, lo que simboliza un **conflicto**.

Algoritmo SLR(1)

@Nov 17, 2020 8:45 AM-10:30 AM

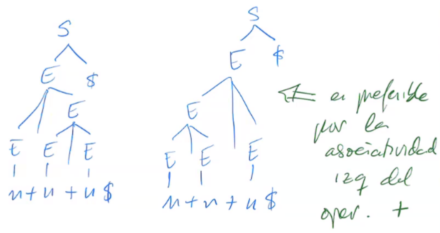
Uso de gramáticas ambigüas

📖 **Gramática ambigua:** para una misma sentencia existe al menos 2 árboles sintácticos diferentes.

Ejemplos:

Sea la gramática G definida por las reglas de abajo.

$$S \to E\$$$
$$E \to E + E$$
$$E \to E * E$$
$$E \to n$$
$$E \to (E)$$



Ambos árboles poseen la misma longitud

En este caso, ninguno de los árboles es mejor que el otro, pero si se le asignara una prioridad como a la de los operadores (la multiplicación antes que la suma), podría decirse que es mejor el izquierdo.

Sea la gramática G definida por las reglas de abajo, realizar la tabla de acciones.

$S \to E\$ \qquad E \to E + E \qquad E \to E * E \qquad E \to n \qquad E \to (E)$

I_0 = { S → ·E$, E → ·E+E , E → ·E*E , E →·n , E → ·(E) }

I_1 = { S → E·$, E → E·+E , E → E·*E }

I_4 = { E → E+·E , E → ·E+E , E → ·E*E , E → ·n , E → ·(E) }

I_7 = { E → E+E· , E → E·+E , E → E·*E }

$I_?$ = { E → E+·E , E → ·E+E , E → ·E*E , E → ·n , E → ·(E) } = I_4 ← Ya aparece

$I_?$ = { E → E*·E , E → ·E+E , E → ·E*E , E → ·n , E → ·(E) } = I_5 ← Ya aparece

I_8 = { E → E+·E }

I_5 = { E → E*·E , E → ·E+E , E → ·E*E , E → ·n , E → ·(E) }

I_2 = { E → n· }

I_3 = { E → (·E) , E → ·E+E , E → ·E*E , E → ·n , E → ·(E) }

I_6 = { E → (E·) , E → E·+E , E → E·*E }

I_7 = { E → n· } = I_2 ← Ya aparece

$\mathrm{TablaTransiciones}(I, X)$ **superpuesta con** $\mathrm{TablaAcciones}(I, X)$:

	n	(	)	+	*	$	E
0	2 D R3	3 D					1
1				4 D	5 D	A	
2			R3	R3	R3	R3	
3	2 D	3 D R3					6

4	2 D	3 D					7
5	2 D	3 D					
6			9 D R4	4 D	5 D		
7			R1	4 D R1	5 D R1	R1	
8			R2	4 D R2	5 D R2	R2	
9			R4	R4	R4	R4	

Fase	Pila	Lexema
1	[0]	• `|` `n` `|+|n|*|n|$|`
2	[0 n 2]	• `|` `n` `|+|n|*|n|$|`
3	[0 E 1]	• `|` `n` `|+|n|*|n|$|`
4	[0 E 1 + 4]	• `|n|` `+` `|n|*|n|$|`
5	[0 E 1 + 4 n 2]	• `|n|+|` `n` `|*|n|$|`
6	[0 E 1 + 4 E]	• `|n|+|n|` `*` `|n|$|`
7	[0 E 1 + 4 E 7]	• `|n|+|n|` `*` `|n|$|`
8 a	[0 E 1]	• `|n|+|n|` `*` `|n|$|`
9 a	[0 E 1 * 5]	• `|n|+|n|` `*` `|n|$|`
10 a	[0 E 1 * 5] n	• `|n|+|n|*|` `n` `|$|`
11 a	[0 E 1 * 5 n 2]	• `|n|+|n|*|n|` `$` `|`
12 a	[0 E 1 * 5 n E 8]	• `|n|+|n|*|n|` `$` `|`
13 a	[0 E 1]	• `|n|+|n|*|n|` `$` `|`
14 a	[ ]	• `|n|+|n|*|n|$|`
8 b	[0 E 1 + 4 E 7 * 5]	• `|n|+|n|*|` `n` `|$|`
9 b	[0 E 1 + 4 E 7 * 5 n 2]	• `|n|+|n|*|n|` `$` `|`
10 b	[0 E 1 + 4 E 7 * 5 E 8]	• `|n|+|n|*|n|` `$` `|`
11 b	[0 E 1 + 4 E 7]	• `|n|+|n|*|n|` `$` `|`

12 b	[0 E 1]	• `|n|+|n|*|n|$|`

🧠 Los conflictos pueden resolverse atendiendo a las preferencias de unos árboles respecto a otros en el caso de gramáticas ambiguas.