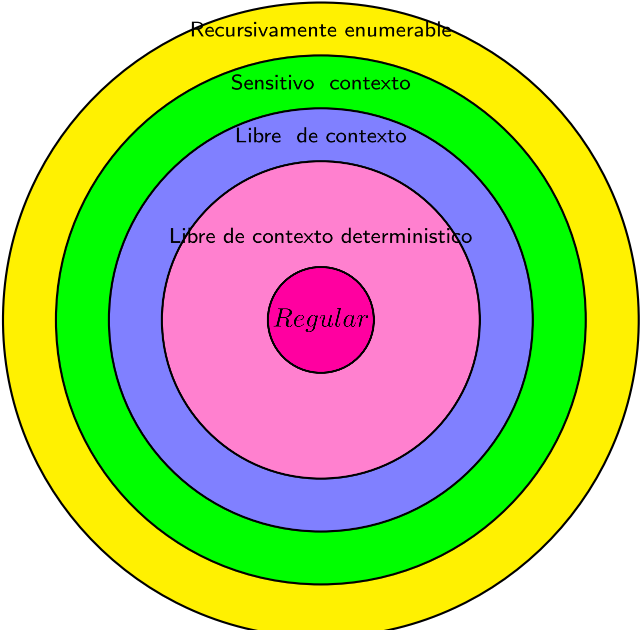
Teoría de Lenguajes
Teórica 11
Gramáticas LL(k) y Parsing LL(1)

Primer Cuatrimestre 2024

Bibliografía para esta clase:
A. V. Aho, J. D. Ullman, The Theory of Parsing, Translation, and Compiling, Vol. 1 , Parsing. Prentice Hall, 1972.
`https://www-2.dc.uba.ar/staff/becher/Aho-Ullman-Parsing-V1.pdf`
Capítulos 4.1, 5.1 y 5.2.

Jeraquía de Lenguajes Formales



Lenguajes delibre de contexto determinísticos

Se llaman lenguajes libres de contexto de determinísticos a los lenguajes reconocibles por autómatas de pila determinísticos. Estos lenguajes son exactamente los lenguajes generables mediante las gramáticas LR, que son una subclase propia de gramáticas libres de contexto.

Un subconjunto propio de lenguajes libres de contexto determísticos son los generables mediante de gramáticas LL. Los estudiaremos hoy.

Gramáticas LL y LR

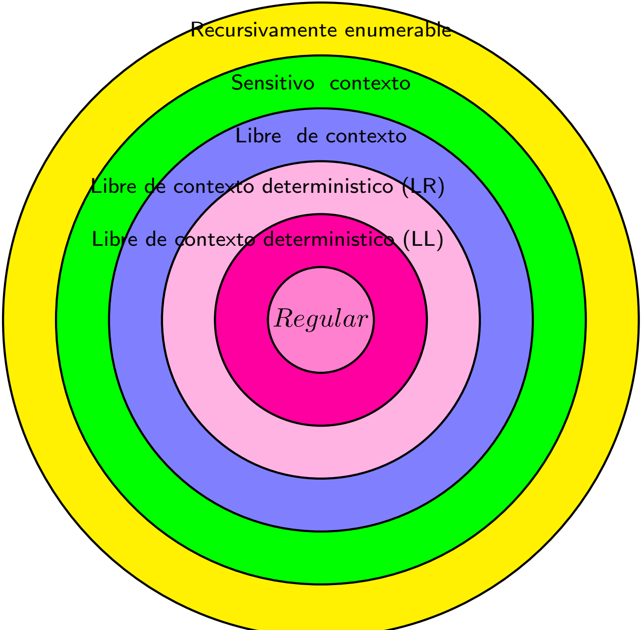
Hay gramáticas libres de contexto que generan lenguajes que se pueden analizar sintácticamente de manera determinística en tiempo lineal en el tamaño de la entrada, leyendo la entrada de izquierda a derecha.

Son las gramáticas LL y LR, los nombres vienen de:

L: left-to right parsing
L: leftmost derivation

L: left-to right parsing
R: rightmost derivation

Jeraquía de Lenguajes Formales



Gramáticas $LL(k)$

Una gramática es LL si es $LL(k)$, para algún número k entero mayor o igual que 1.

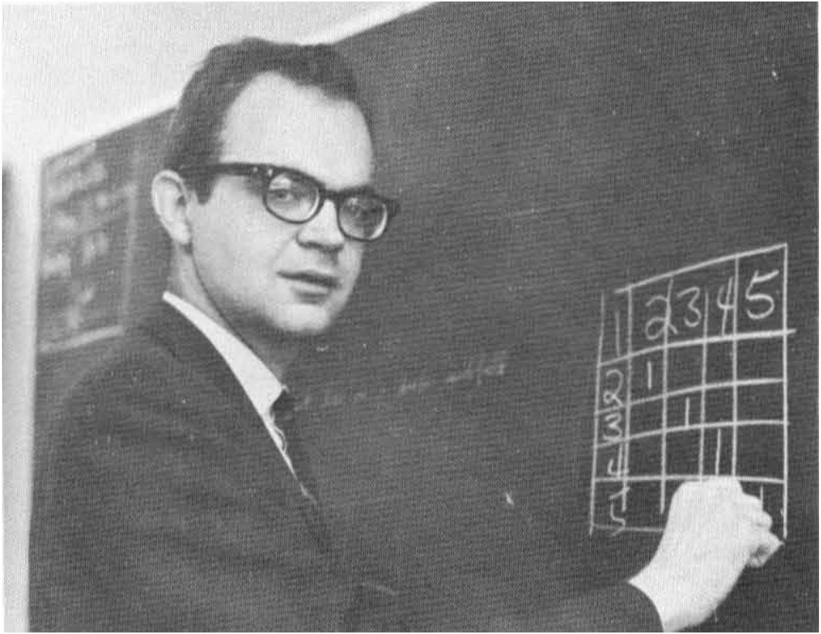
Las gramáticas $LL(k)$ son gramaticas libres de contexto para las cuales la derivación más a la izquierda está determinada por los símbolos ya leídos, y k símbolos más.

El parsing asociado es "top down", es decir, va encontrando una tras otra, las producciones que conforman la derivación que empieza con el símblo de inicio S hasta la cadena. Cada paso de la derivación se resuelve esencialmente en tiempo constante, y se demuestra que el tiempo total es lineal en el tamaño de la entrada.

Para todo lenguaje $LL(k)$ hay un autómata de pila determinístico con un solo estado que lo reconoce.

Korenjack y Hopcroft definieron las gramaticas $LL(1)$ en 1966.
Lewis y Stearns (1968), Knuth (1967) y otros, hicieron la teoría y los algoritmos para $LL(k)$.

Donald Knuth 1965



The Art of Computer Programming

Gramáticas LL(k)

Sea $G = (T, N, P, S)$ libre de contexto y sea $w \in T^*$ una cadena de $L(G)$. Entonces hay una única secuencia $\alpha_0, \alpha_1, \ldots, \alpha_m$ de formas sentenciales tal que

$$S = \alpha_0, \quad \alpha_i \underset{L}{\Rightarrow} \alpha_{i+1}, \text{ para } i = 0, 1, \ldots, m-1, \text{ y } \alpha_m = w$$

El parsing a izquierda de w es la secuencia de las m producciones usadas en la derivación de w.

Las gramáticas $LL(k)$ son tales que si $\alpha_i = a_1 \ldots a_j A\beta$ entonces α_{i+1} es determinable conociendo solamente $a_1 \ldots a_j$ y k símbolos más del input $a_{j+1} \ldots a_{j+k}$ para el k que hemos fijado.

Ejemplo de gramática $LL(k)$

$$
\begin{array}{rcl}
S & \rightarrow & cAd \\
A & \rightarrow & ab|a
\end{array}
$$

Consideremos $w = aa$. Vemos $S \underset{L}{\overset{*}{\not\Rightarrow}} aa$.
Consideremos $w = cad$

$$
\begin{array}{rcl}
S \underset{L}{\Rightarrow} cAd & \underset{L}{\Rightarrow} & c\underbrace{ab}d \\
S \underset{L}{\Rightarrow} cAd & \underset{L}{\Rightarrow} & c\underbrace{\ a\ }d
\end{array}
$$

Los dos símbolos que nos tocan leer de la entrada, junto con el no-terminal más a la izquierda, determinan cuál de las tres producciones debemos usar.
Esta gramática es $LL(2)$.

Ejemplo de gramática $LL(1)$

$$\begin{array}{lcl} S & \rightarrow & aAS|b \\ A & \rightarrow & a|bSA \end{array}$$

Sea $w = aaaab$.

$$S \underset{L}{\Rightarrow} aAS \underset{L}{\Rightarrow} aaS \underset{L}{\Rightarrow} aaaAS \underset{L}{\Rightarrow} aaaaS \underset{L}{\Rightarrow} aaaab$$

El símbolo que nos toca leer de la entrada, junto con el no-terminal más a la izquierda, determinan cuál de las cuatro producciones debemos usar.

Ejemplo de gramática que no es $LL(k)$ para ningun k

$$
\begin{aligned}
S &\to A|B \\
A &\to aAb|0 \\
B &\to aBbb|1
\end{aligned}
$$

$L(G) = \{a^n 0 b^n : n \geq 0\} \cup \{a^n 1 b^{2n} : n \geq 0\}$

Notemos que para todo valor de $k \geq 1$,

$$
\begin{aligned}
S &\underset{L}{\Rightarrow} A \overset{*}{\underset{L}{\Rightarrow}} a^k 0 b^k \\
S &\underset{L}{\Rightarrow} B \overset{*}{\underset{L}{\Rightarrow}} a^k 1 b^{2k}
\end{aligned}
$$

Los primeros k símbolos en $a^k 0 b^k$y $a^k 1 b^{2k}$ coinciden, por lo cual no determinan qué producción hay que usar.

Gramáticas $LL(k)$

Definición (PRIMEROS$_k(\alpha)$)

Sea $k \geq 1$. Dada una gramática libre de contexto $G = \langle N, T, P, S \rangle$.
Definimos PRIMEROS$_k$: $(T \cup N)^* \rightarrow \mathcal{P}(T)$

$$\textit{PRIMEROS}_k(\alpha) = \left\{ z \in T^+ : \begin{array}{l} |z| < k, \alpha \overset{*}{\underset{L}{\Rightarrow}} z, \text{ ó} \\ |z| = k, \alpha \overset{*}{\underset{L}{\Rightarrow}} zw \text{ para alguna } w \in T^* \end{array} \right\}$$

En particular para $k = 1$,

$$\text{PRIMEROS}(\alpha) = \left\{ z \in T : z = a \in T, \alpha \overset{*}{\underset{L}{\Rightarrow}} aw \text{ para alguna } w \in T^* \right\}$$

Ejemplo de PRIMEROS$_k(\alpha)$

$$
\begin{aligned}
S &\to A|B \\
A &\to aAb|0 \\
B &\to aBbb|1
\end{aligned}
$$

$$
\begin{aligned}
S \quad &\overset{*}{\underset{L}{\Rightarrow}} \quad S \underset{L}{\Rightarrow} A \overset{*}{\underset{L}{\Rightarrow}} a^k 0 b^k \\
S \quad &\overset{*}{\underset{L}{\Rightarrow}} \quad S \underset{L}{\Rightarrow} B \overset{*}{\underset{L}{\Rightarrow}} a^k 1 b^{2k}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{PRIMEROS}(a) &= \{a\} \\
\mathsf{PRIMEROS}(aA) &= \{a\} \\
\mathsf{PRIMEROS}(S) &= \{0, 1, a\} \\
\mathsf{PRIMEROS}_2(S) &= \{0, 1, a0, a1, aa\} \\
\mathsf{PRIMEROS}_3(S) &= \{0, 1, a0b, a1b, aa0, aa1, aaa\} \\
\mathsf{PRIMEROS}_k(S) &= \{0, 1, \ldots, a^k\} \\
\mathsf{PRIMEROS}(A) &= \{0, a\} \\
\mathsf{PRIMEROS}_2(A) &= \{0, a0, aa\} \\
\mathsf{PRIMEROS}_3(A) &= \{0, a0b, aa0, aaa\} \\
\mathsf{PRIMEROS}_k(A) &= \{0, \ldots, a^k\}
\end{aligned}
$$

Gramática $LL(k)$

Definición (Gramática $LL(k)$)

Una gramática libre de contexto $G = \langle N, T, P, S \rangle$ es $LL(k)$ si ocurre que NO hay dos derivaciones más a la izquierda tales que

$$\begin{array}{rcl} S \overset{*}{\underset{L}{\Rightarrow}} wA\alpha & \underset{L}{\Rightarrow} & w\beta\alpha \overset{*}{\underset{L}{\Rightarrow}} wx \\ S \overset{*}{\underset{L}{\Rightarrow}} wA\alpha & \underset{L}{\Rightarrow} & w\gamma\alpha \overset{*}{\underset{L}{\Rightarrow}} wy \\ \mathit{PRIMEROS}_k(x) & = & \mathit{PRIMEROS}_k(y) \\ \beta & \neq & \gamma \end{array}$$

Una gramática que no es LL

$$
\begin{aligned}
S &\to A|B \\
A &\to aAb|0 \\
B &\to aBbb|1
\end{aligned}
$$

$L(G) = \{a^n 0 b^n : n \geq 0\} \cup \{a^n 1 b^{2n} : n \geq 0\}$

Notemos que para todo valor de $k \geq 0$,

$$
S \underset{L}{\overset{*}{\Rightarrow}} S \quad \underset{L}{\Rightarrow} \quad \underbrace{A}_{\beta} \underset{L}{\overset{*}{\Rightarrow}} a^k 0 b^k
$$

$$
S \underset{L}{\overset{*}{\Rightarrow}} S \quad \underset{L}{\Rightarrow} \quad \underbrace{B}_{\gamma} \underset{L}{\overset{*}{\Rightarrow}} a^k 1 b^{2k}
$$

$$
\text{PRIMEROS}_k(a^k 0 b^k) \quad = \quad \text{PRIMEROS}_k(a^k 1 b^{2k}) = a^k,
$$

pero $A = \beta$, $B = \gamma$, $\beta \neq \gamma$.
Por lo tanto G no es $LL(k)$, para ningún k.

Las gramáticas LL no son ambiguas

Teorema

Toda gramática LL es no ambigua.

Demostración. Supongamos una gramática $LL(k)$ que es ambigua. Entonces hay una cadena w y dos derivaciones distintas

$$S \underset{L}{\Rightarrow} \alpha_1 \underset{L}{\Rightarrow} \alpha_2 \ldots \underset{L}{\Rightarrow} \alpha_n \underset{L}{\Rightarrow} w$$

$$S \underset{L}{\Rightarrow} \beta_1 \underset{L}{\Rightarrow} \beta_2 \ldots \underset{L}{\Rightarrow} \beta_m \underset{L}{\Rightarrow} w$$

Sea i el mínimo tal que $\alpha_i \neq \beta_i$. Entonces, $\alpha_{i-1} = \beta_{i-1}$ y por lo tanto α_{i-1} y β_{i-1} tienen el mismo no-terminal más a la izquierda. Tienen forma $xA\alpha$. Como ambas derivaciones llegan a w, sabemos que $w = xy$.

$$\begin{array}{rcl} S \overset{*}{\underset{L}{\Rightarrow}} \alpha_{i-1} & \underset{L}{\Rightarrow} & \alpha_i \overset{*}{\underset{L}{\Rightarrow}} w \\ S \overset{*}{\underset{L}{\Rightarrow}} \beta_{i-1} & \underset{L}{\Rightarrow} & \beta_i \overset{*}{\underset{L}{\Rightarrow}} w \\ y & = & y \\ \alpha_i & \neq & \beta_i . \end{array}$$

Esto contradice que la gramática es $LL(k)$. $\square$

Gramáticas sin ciclos

Definición (Gramática sin ciclos)

Una gramática no tiene ciclos si para todo símbolo no terminal A no hay derivaciones $A \stackrel{*}{\Rightarrow} A$.

En las gramáticas libre de contexto los ciclos se originan en la existencia de producciones $A \rightarrow B$ con B un único no-terminal y en las producciones $A \rightarrow \lambda$.
Es posible transformar la gramática en otra sin ciclos eliminando las producciones $A \rightarrow \lambda$ (solamente podemos dejar $S \rightarrow \lambda$) y eliminando las producciones con un solo no-terminal en el cuerpo.
Ver Algoritmos 2.10 y 2.11 [Aho Ullman vol. 1].

Ciclos y recursion a izquierda implican ambiguedad

Supongamos G es recursiva a izquierda en A y además tiene ciclo para A. Entonces G es ambigua, ya que hay dos derivaciones distintas para la misma expresión

$$
\begin{aligned}
A \underset{L}{\overset{*}{\Rightarrow}} A\alpha \quad & \underset{L}{\overset{*}{\Rightarrow}} \quad A\alpha \\
A \underset{L}{\overset{*}{\Rightarrow}} A \quad & \underset{L}{\overset{*}{\Rightarrow}} \quad A\alpha
\end{aligned}
$$

Símbolo alcanzable y símbolo activo

Definición (Símbolo alcanzable)

Dada una gramática $G = \langle N, T, P, S \rangle$ *el no-terminal* A *es alcanzable si para* $\alpha, \beta \in (T \cup N)^*$,

$$S \overset{*}{\underset{G}{\Rightarrow}} \alpha A \beta$$

Definición (Símbolo activo)

Dada una gramática $G = \langle N, T, P, S \rangle$ *el no-terminal* A *es activo si para algún* $w \in T^*$,

$$A \overset{*}{\underset{G}{\Rightarrow}} w$$

Las gramáticas LL no son recursivas a izquierda

Teorema

Toda gramática $LL(k)$ sin ciclos y en la cual todos los no-terminales son alcanzables y activos no es recursiva a izquierda.

Demostración

Fijemos k. Para simplificar asumimos un nuevo símbolo $\perp$ y un nuevo símbolo no-terminal S_k con una produccion $S_k \to S\perp^k$.

Supongamos que sí es recursiva a izquierda, por lo tanto hay $A \to \gamma$, $A \to B\beta$, con $\gamma \neq B\beta$ y $B\beta \overset{*}{\Rightarrow} A\alpha$. Por lo tanto $A \overset{*}{\Rightarrow} A\alpha$.

Entonces,

$$S_k \overset{*}{\Rightarrow} xA... \Rightarrow xB\beta.... \overset{*}{\Rightarrow} xA\alpha^k... \Rightarrow x \underbrace{\gamma} \alpha^k... \overset{*}{\Rightarrow} xz...$$

$$S_k \overset{*}{\Rightarrow} xA... \Rightarrow xB\beta.... \overset{*}{\Rightarrow} xA\alpha^k... \Rightarrow x \underbrace{B\beta} \alpha^k.... \overset{*}{\Rightarrow} xz...$$

ya que $xB\beta\alpha^k....\Rightarrow xA\alpha^{k+1}\Rightarrow x\gamma\alpha^{k+1}.... \overset{*}{\Rightarrow} xz...$

Obviamente $z = z$, y $|z| \geq k$
Pero $\gamma \neq B\beta$
Entonces la gramática no es $LL(k)$. $\square$

Teorema sobre gramáticas $LL(k)$

Teorema

Una gramática libre de contexto $G = \langle N, T, P, S \rangle$ *es* $LL(k)$ *si y solo si, para todos los* $wA\alpha$ *tales que* $S \overset{*}{\underset{L}{\Rightarrow}} wA\alpha$ *y para todo par de producciones* $A \to \beta$ *y* $A \to \gamma$, *con* $\beta \neq \gamma$, *PRIMEROS*$_k(\beta\alpha) \cap$ *PRIMEROS*$_k(\gamma\alpha) = \emptyset$,

Ejemplo

$$\begin{array}{rcl} S & \rightarrow & cAa \\ A & \rightarrow & \lambda | a \end{array}$$

Supongamos $w = caa$

Entonces $S \underset{L}{\Rightarrow} cAa$

Son disjuntos:
$PRIMEROS(a) = \{a\}$
$PRIMEROS(\lambda) = \{\}$

Sin embargo, estos no lo son:
$PRIMEROS(\lambda a) = \{a\}$
$PRIMEROS(aa) = \{a\}$

Demostración del Teorema

Supongamos $S \overset{*}{\underset{L}{\Rightarrow}} wA\alpha$ y producciones $A \to \beta$, $A \to \gamma$ con $\beta \neq \gamma$ y

$$\begin{aligned} S &\overset{*}{\underset{L}{\Rightarrow}} wA\alpha \underset{L}{\Rightarrow} w\beta\alpha \overset{*}{\underset{L}{\Rightarrow}} wxy \\ S &\overset{*}{\underset{L}{\Rightarrow}} wA\alpha \underset{L}{\Rightarrow} w\gamma\alpha \overset{*}{\underset{L}{\Rightarrow}} wxz, \end{aligned}$$

con $x \in \text{PRIMEROS}_k(\beta\alpha) \cap \text{PRIMEROS}_k(\gamma\alpha)$ tal que si $|x| < k$ entonces $y = z = \lambda$. Como $\beta \neq \gamma$, G no es $LL(k)$.

Supongamos ahora que G no es $LL(k)$ entonces existen dos derivaciones

$$\begin{aligned} S &\overset{*}{\underset{L}{\Rightarrow}} wA\alpha \underset{L}{\Rightarrow} w\beta\alpha \overset{*}{\Rightarrow} wx \\ S &\overset{*}{\underset{L}{\Rightarrow}} wA\alpha \underset{L}{\Rightarrow} w\gamma\alpha \overset{*}{\Rightarrow} wy \end{aligned}$$

y existe $z \in T^*$, $|z| \leq k$ tal que $z = \text{PRIMEROS}_k(x) = \text{PRIMEROS}_k(y)$.
Entonces, $z \in \text{PRIMEROS}_k(\beta\alpha)$ y $z \in \text{PRIMEROS}_k(\gamma\alpha)$.
Por lo tanto, $\text{PRIMEROS}_k(\beta\alpha) \cap \text{PRIMEROS}_k(\gamma\alpha) \neq \emptyset$. $\square$

Algoritmo de parsing $LL(1)$

Sea $G = (N, T, P, S)$ libre de contexto. Escribimos letras griegas para expresiones en $(N \cup T)^*$, e imprenta mayúscula para no-terminales. Necesitamos tres definiciones :

PRIMEROS: $(N \cup T)^* \to \mathcal{P}(T)$,
PRIMEROS$(\alpha) = \{a \in T : \alpha \underset{L}{\overset{*}{\Rightarrow}} a\beta\}$

SIGUIENTES: $N \to \mathcal{P}(T) \cup \{\$\}$,
SIGUIENTES$(A) = \{a \in T : S \underset{L}{\overset{*}{\Rightarrow}} \alpha A \beta, a \in$ PRIMEROS$(\beta)\} \cup \{\$: S \underset{L}{\overset{*}{\Rightarrow}} \alpha A\}$

$SD : P \to \mathcal{P}(T)$

$$SD(A \to \beta) = \left\{ \begin{array}{l} \text{PRIMEROS}(\beta), \text{ si } \beta \text{ no es anulable} \\ \text{PRIMEROS}(\beta) \cup \text{SIGUIENTES}(A), \text{ si } \beta \text{ es anulable} \end{array} \right\}$$

Un ejemplo

$$
\begin{aligned}
S &\to AaAb|BbBa \\
A &\to \lambda \\
B &\to \lambda
\end{aligned}
$$

$$
\begin{aligned}
\textsf{PRIMEROS}(S) &= \textsf{PRIMEROS}(A) \cup \textsf{PRIMEROS}(a) \\
&\quad \cup \textsf{PRIMEROS}(B) \cup \textsf{PRIMEROS}(b) \\
&= \{a, b\} \\
\textsf{PRIMEROS}(A) &= \{\} \\
\textsf{PRIMEROS}(B) &= \{\} \\
\textsf{SIGUIENTES}(S) &= \{\$\} \\
\textsf{SIGUIENTES}(A) &= \textsf{PRIMEROS}(a) \cup \textsf{PRIMEROS}(b) = \{a, b\} \\
\textsf{SIGUIENTES}(B) &= \textsf{PRIMEROS}(b) \cup \textsf{PRIMEROS}(a) = \{a, b\}
\end{aligned}
$$

Tabla $LL(1)$

Sea $G = (N, T, P, S)$.

$$SD(A \to \beta) = \begin{cases} \text{PRIMEROS}(\beta), \text{ si } \beta \text{ no es anulable} \\ \text{PRIMEROS}(\beta) \cup \text{SIGUIENTES}(A), \text{ si } \beta \text{ es anulable} \end{cases}$$

La tabla $LL(1)$ es una matriz M de dimensión $|N| \times |T \cup \{\$\}|$

$$M(A, a) = A \to \beta \text{ si y solo si } a \in SD(A \to \beta)$$

Ejemplo.

$$\begin{aligned} S &\to AB \\ A &\to aA|ca \\ B &\to bB|cb \end{aligned}$$

$$\begin{aligned} &SD(A \to aA) = \{a\} \\ &SD(A \to ca) = \{c\} \\ &SD(B \to bB) = \{b\} \\ &SD(B \to cb) = \{c\} \\ &SD(S \to AB) = \{a, c\} \end{aligned}$$

	a	b	c	$\$$
S	$S \to AB$		$S \to AB$	
A	$A \to aA$		$A \to ca$	
B		$B \to bB$	$B \to cb$	

Tabla para algoritmo $LL(1)$

Input $G = (N, T, P, S)$
Output Matriz M dimensión $|N| \times (|T \cup \{\$\}|)$
for cada producción $A \to \alpha \in P$ **do**
    for cada $a \in$ PRIMEROS(α) **do**
        $M[A, a] \leftarrow M[A, a] \cup \{A \to \alpha\}$
    end for
    if α es anulable **then**
        for cada $b \in$ SIGUIENTES(A) **do**
            $M[A, b] \leftarrow M[A, b] \cup \{A \to \alpha\}$
        end for
        if $\$ \in$ SIGUIENTES(A) **then**
            $M[A, \$] \leftarrow M[A, \$] \cup \{A \to \alpha\}$
        end if
    end if
end for
for toda posición $M[A, a] == \varnothing$ **do**
    $M[A, a] \leftarrow \mathbf{error}$
end for

Ejemplo corrida algoritmo $LL(1)$

$$
\begin{aligned}
S &\rightarrow AB \\
A &\rightarrow aA|ca \\
B &\rightarrow bB|cb
\end{aligned}
$$

	a	b	c	$\$$
S	$S \rightarrow AB$		$S \rightarrow AB$	
A	$A \rightarrow aA$		$A \rightarrow ca$	
B		$B \rightarrow bB$	$B \rightarrow cb$	

Input $cacb\$$

Pila	Input	Acción
$\$S$	$cacb\$$	$print\ S \rightarrow AB$
$\$BA$	$cacb\$$	$print\ A \rightarrow ca$
$\$Bac$	$cacb\$$	Desapilar
$\$Ba$	$acb\$$	Desapilar
$\$B$	$cb\$$	$print\ B \rightarrow cb$
$\$bc$	$cb\$$	Desapilar
$\$b$	$b\$$	Desapilar
$\$$	$\$$	Termina

$$S \underset{L}{\Rightarrow} AB \underset{L}{\Rightarrow} caB \underset{L}{\Rightarrow} cacb$$

Algoritmo parsing $LL(1)$

Input Matriz M, cadena $w\$$
Output Lista de producciones de derivación $S \overset{*}{\underset{L}{\Rightarrow}} w$.

Comenzar con la pila en $\$S$ (es decir, S en el tope)
Puntero de entrada en la primera posición de $w\$$
repeat
  Sea a símbolo apuntado en la cadena de entrada $w\$$
  Sea X el tope de pila
  if $X \in (T \cup \{\$\})$ **then**
    if $X == a$ **then**
      Desapilar X y avanzar el puntero de la cadena de entrada
    else Reportar **error**
    end if
  else
    if $M[X, a] == X \to Y_1 Y_2 \ldots Y_k$ **then**
      Desapilar X
      Apilar $Y_k Y_{k-1} \ldots Y_1$ (con Y_1 en el tope)
      Emitir la producción $X \to Y_1 Y_2 \ldots Y_k$
    else Reportar **error**
    end if
  end if
until $X == \$$

Complejidad del algoritmo $LL(1)$

Teorema (Teorema 5.6 Aho Ullman Vol 1)

El algoritmo $LL(1)$ *realiza una cantidad de pasos lineal en el tamaño de la entrada.*

Demostración

La cantidad de pasos que realiza el agoritmo para aceptar un cadena w surge de la cantidad de iteraciones del ciclo. Dado que cada iteración se desapila un símbolo, terminal o no-terminal, contaremos la cantidad de veces que se desapila.
Supongamos que la cadena de entrada w tiene n símbolos. El algoritmo requiere que cada uno de estos n símbolos terminales sean desapilados.

¿Cuántas iteraciones hay entre dos veces que se desapila un símbolo terminal? Dado que G es $LL(1)$, no es recursiva a izquierda, por lo tanto no hay derivaciones $A \underset{L}{\overset{+}{\Rightarrow}} A\alpha$. Necesitamos acotar el número de pasos en las derivaciones de la forma $A \underset{L}{\overset{+}{\Rightarrow}} B\alpha$, con $A \neq B$. El lema que sigue (Lema 4.1 Aho-Ullman vol. 1) afirma que este número de pasos está acotado por una constante c. Concluímos que el algoritmo desapila a lo sumo $(c+1)n$ veces. □

Lema: cota en la cantidad de pasos en una derivación

Lema (Lema 4.1 Aho-Ullman vol. 1)

Sea $G = (N, T, P, S)$ libre de contexto y no recursiva. Existe una constante c tal que si $A \overset{i}{\underset{L}{\Rightarrow}} wB\alpha$ y $|w| = n$ entonces $i \leq c^{n+2}$.

Se puede demostrar un resultado mucho más ajustado, con i lineal en n, pero la misma constante.

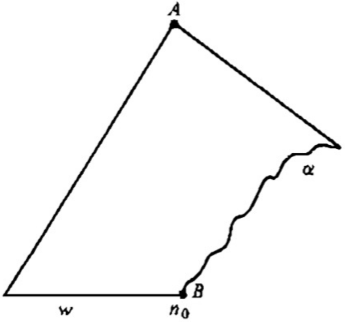
En particular, el resultado anterior vale para w igual a la cadena vacía. Así obtenemos el resultado que necesitamos para la demostración del teorema.

Corolario

Sea G libre de contexto y no recursiva. Existe una constante c tal que para todo par de no terminales A, B, si $A \overset{i}{\underset{L}{\Rightarrow}} B\alpha$ entonces $i \leq c^2$.

Demostración del lema

Llamemos k a la cantidad de símboos no-terminales.
Sea $\mathcal{A}$ el ábol de la derivación más a la izquierda para $A \overset{i}{\underset{L}{\Rightarrow}} wB\alpha$.



Sea n_0 el nodo con etiqueta B en la derivación $A \overset{i}{\underset{L}{\Rightarrow}} wB\alpha$. Notemos que, por tratarse de la derivación más a la izquierda, todos los caminos a la derecha del camino desde la raíz a n_0 son más cortos, o del mismo largo.

Supongamos que hay un camino de longitud mayor o igual que $k(n+2)$ arcos de la raíz a la hoja,

$$A = \alpha_0 \underset{L}{\Rightarrow} \alpha_1 \underset{L}{\Rightarrow} \ldots \underset{L}{\Rightarrow} \alpha_{k(n+2)-1} \underset{L}{\Rightarrow} \alpha_{k(n+2)} = wB\alpha$$

Visualicemos esta derivación por segmentos así:

$$\alpha_0 \underset{L}{\Rightarrow} \alpha_1 \underset{L}{\Rightarrow} \ldots \underset{L}{\Rightarrow} \alpha_k$$

$$\alpha_k \underset{L}{\Rightarrow} \ldots \underset{L}{\Rightarrow} \alpha_{2k}$$

$$\ldots$$

$$\alpha_{(n+1)k} \underset{L}{\Rightarrow} \ldots \underset{L}{\Rightarrow} \alpha_{(n+2)k}.$$

Son $(n+2)$ segmentos de derivaciones. Es imposible que cada uno de estos produzca uno o más símbolos de wB, porque $|wB| = n+1$. Entonces ¡hay al menos uno de estos segmentos que no produce ningun símbolo!

Entonces en el árbol de derivación $\mathcal{A}$ hay un segmento, digamos el iésimo,

$$\alpha_{ik} \underset{L}{\Rightarrow} \dots \underset{L}{\Rightarrow} \alpha_{(i+1)k}$$

que no produce ningún símbolo de wB.

Llamemos $v_0, .., v_k$ a los vértices cuyas etiquetas son $\alpha_{ik}, .., \alpha_{(i+1)k}$. Entonces, el subarbol $v_0, .., v_k$ deriva solamente λ. Como cada uno de $\alpha_{ik}, .., \alpha_{(i+1)k}$ es una cadena de símbolos no terminales y son en total $k+1$, entonces, necesariamente hay dos que empiezan con el mismo símbolo. Pero esto contradice que la gramática no es recursiva a izquierda. Entonces nuestra suposición de que el árbol de derivación $\mathcal{A}$ tiene un camino de longitud mayor igual que $k(n+2)$ es imposible.

Sea ℓ el máximo número de símbolos en la parte derecha de una producción de la gramática. La cantidad de nodos del árbol de derivación $\mathcal{A}$ es a lo sumo

$$\ell^{k(n+2)}$$

Por lo tanto, si $A \overset{i}{\underset{L}{\Rightarrow}} wB\alpha$, entonces $i \leq \ell^{k(n+2)}$.

Para finalizar la demostración basta tomar $c = \ell^k$. $\square$