Teoría de Lenguajes
Segunda parte
Teórica 10: Parsing $LL(1)$

Verónica Becher

Segundo cuatrimestre 2020

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.

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. Tenemos las definiciones

$$\begin{aligned}
\text{PRIMEROS}(\alpha) &= \{a \in T \cup \{\lambda\} : \alpha \underset{L}{\overset{*}{\Rightarrow}} \lambda \text{ ó } \alpha \underset{L}{\overset{*}{\Rightarrow}} a\beta\} \\
\text{SIGUIENTES}(A) &= \{a \in T : S \underset{L}{\overset{*}{\Rightarrow}} \alpha A\beta, a \in \text{PRIMEROS}(\beta)\} \\
&\quad \cup \{\$: S \underset{L}{\overset{*}{\Rightarrow}} \alpha A\} \\
SD(A \to \beta) &= \left\{ \begin{array}{l} \text{PRIMEROS}(\beta), \text{ si } \beta \text{ no es anulable} \\ (\text{PRIMEROS}(\beta) \setminus \{\lambda\}) \cup \text{SIGUIENTES}(A), \text{ si } \beta \text{ es anulable} \end{array} \right\}
\end{aligned}$$

Un ejemplo

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

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

Otro ejemplo

$$
\begin{aligned}
S &\to (L)|a \\
L &\to SM \\
M &\to !SM|\lambda \\
\text{PRIMEROS}(S) &= \{(, a\} \\
\text{PRIMEROS}(L) &= \text{PRIMEROS}(S) = \{(, a\} \\
\text{PRIMEROS}(M) &= \{!, \lambda\}
\end{aligned}
$$

$$
\begin{aligned}
\text{SIGUIENTES}(S) &= \{\$\} \\
&\cup (\text{PRIMEROS}(M) \setminus \{\lambda\}) \\
&\cup \text{SIGUIENTES}(L) \\
&\cup \text{SIGUIENTES}(M) \\
&= \{\$, !,)\} \\
\text{SIGUIENTES}(L) &= \{)\} \\
\text{SIGUIENTES}(M) &= \text{SIGUIENTES}(L) = \{)\}
\end{aligned}
$$

Tabla $LL(1)$

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

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

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.

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

$SD(A \to aA) = \{a\}$
$SD(A \to cA) = \{c\}$
$SD(B \to bB) = \{b\}$
$SD(S \to AB) = \{a, c\}$

	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 $a \in \text{PRIMEROS}(\alpha)$ **do**
    $M[A, a] \leftarrow M[A, a] \cup \{A \to \alpha\}$
  end for
  if $\lambda \in \text{PRIMEROS}(\alpha)$ **then**
    for cada $b \in \text{SIGUIENTES}(A)$ **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**
$M[A, a] \leftarrow$ **error**
end for

Ejemplo corrida algoritmo $LL(1)$

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

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

Input $cacb\$$

Pila	Input	Acción
$\$S$	$cacb\$$	*print* $S \to AB$
$\$BA$	$cacb\$$	*print* $A \to ca$
$\$Bac$	$cacb\$$	Desapilar
$\$Ba$	$acb\$$	Desapilar
$\$B$	$cb\$$	*print* $B \to cb$
$\$bc$	$cb\$$	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 $M[X, a] == X \to Y_1 Y_2 \dots Y_k$ **then**
Desapilar X
Apilar $Y_k Y_{k-1} \dots Y_1$ (con Y_1 en el tope)
Emitir la producción $X \to Y_1 Y_2 \dots Y_k$
else Reportar **error**
end if
end if
until $X == \$$

Complejidad del algoritmo $LL(1)$

Teorema 1 (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 $A \overset{+}{\underset{L}{\Rightarrow}} A\alpha$. Necesitamos acotar el número de pasos en las derivaciones de la forma $A \overset{+}{\underset{L}{\Rightarrow}} B\alpha$, 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 2 (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 $i \leq c^{n+1}$.*

Se puede demostrar un resultado mucho más ajustado, con i lineal en n.

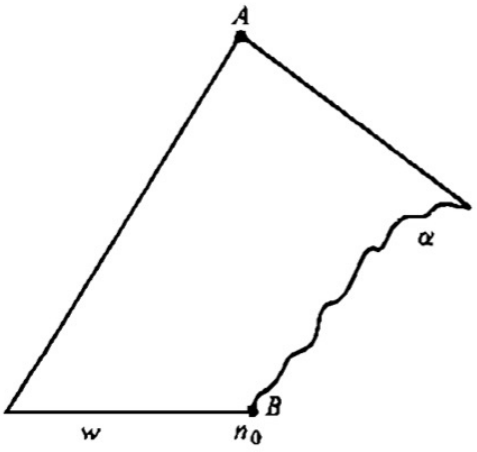
En particular, el resultado anterior vale para w igual a la cadena vacía. Así obtenemos el resultado que necesitamos para la demostración del teorema.

Corolario 3. *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 $i \leq c$.*

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 C de longitud mayor o igual que $(k+1)(n+1)$ nodos de la raíz a la hoja,

$$C = S_1 S_2 \dots S_{n+1},$$

donde cada S es una secuencia de al menos $k+1$ nodos consecutivos.

Como $|wB|$ tiene longitud $n+1$, es imposible que todas las secuencias S arrojen más de un símbolo de wB . Entonces hay una secuencia $S = n_1, \dots, n_{k+1}$, que produce exactamente un símbolo de wB o ninguno. Para cada $j = 1, 2, \dots, k$, todos los descendientes directos de n_j que están a la izquierda de n_{j+1} derivan en λ, es decir, se anulan.

Como cada uno de $n_1, \dots, n_{k+1}$ tiene como etiqueta un símbolo no terminal, hay dos con la misma etiqueta. 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 $(k+1)(n+1)$ 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

$$\ell^{(k+1)(n+1)}$$

Por lo tanto, si $A \underset{L}{\overset{i}{\Rightarrow}} wB\alpha$, entonces $i \leq \ell^{(k+1)(n+1)}$.

Para finalizar la demostración basta tomar $c = \ell^{k+1}$. □