Teoría de Lenguajes
Segunda parte
Teórica 11: Parsing $LR(1)$

Verónica Becher

Primer cuatrimestre 2020

Bibliografía para esta clase:

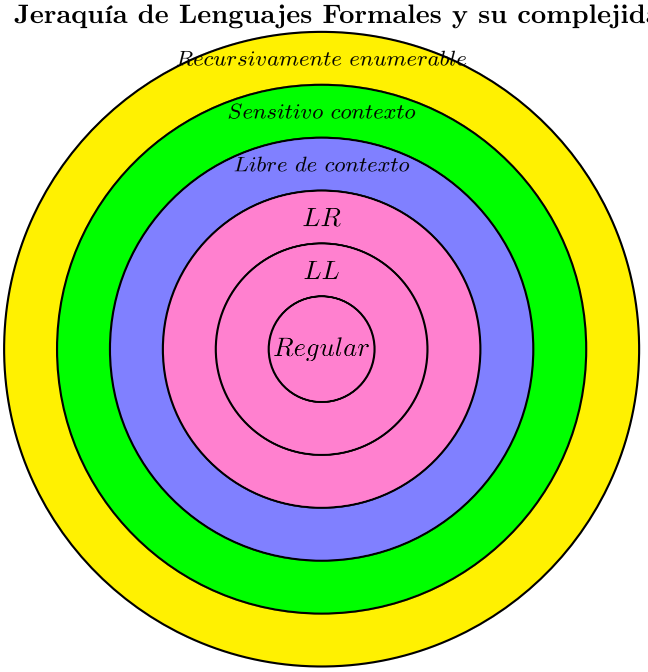
A. V. Aho, J. D. Ullman, The Theory of Parsing, Translation, and Compiling, Vol. 1 , Parsing. Prentice Hall, 1972.

`https://www-2.dc.uba.ar/staff/becher/Aho-Ullman-Parsing-V1.pdf` Capítulos 2, 5.1 y 5.2.

A. V. Aho, Sethi, J. D. Ullman, Segunda edicion , 2006. (el libro del dragon) Compilers: Principles, Techniques, and Tools, Addison-Wesley

`https://www-2.dc.uba.ar/staff/becher/dragon.pdf` Capitulo 4

Jeraquía de Lenguajes Formales y su complejidad de parsing



Complejdad
Rosa: lineal
Azul: cúbica
Verde: PSPACE
Amarillo: PSPACE

Gramáticas $LR(k)$

Definición (Gramática $LR(k)$). *Dada una gramática libre de contexto $G = (N, T, P, S)$ definimos la gramática aumentada $G' = (N \cup \{S'\}, T, P \cup \{S' \to S\}, S')$.*

Decimos que G es $LR(k)$, con $k \geq 0$, si estas tres condiciones

$$S' \underset{R}{\overset{*}{\Rightarrow}} \alpha A w \underset{R}{\Rightarrow} \alpha\beta w$$

$$S' \underset{R}{\overset{*}{\Rightarrow}} \gamma B x \underset{R}{\Rightarrow} \alpha\beta y$$

$$PRIMEROS_k(w) = PRIMEROS_k(y)$$

implican $\alpha = \gamma$, $A = B$ *y* $x = y$, *o sea* $\alpha Ay = \gamma Bx$.

En particular, la gramática G es $LR(0)$ si

$$S' \underset{R}{\overset{*}{\Rightarrow}} \alpha Aw \underset{R}{\Rightarrow} \alpha\beta w$$
$$S' \underset{R}{\overset{*}{\Rightarrow}} \gamma Bx \underset{R}{\Rightarrow} \alpha\beta y$$

implican que $\alpha = \gamma$, $A = B$ y $x = y$, o sea $\alpha Ay = \gamma Bx$.

Lenguajes LR

Teorema 1 (Theorem 8.1 Aho Ulman vol 2). *Para toda gramática* G *que es* $LR(k)$, $k \geq 0$, *hay una gramática* G' *que es* $LR(1)$ *tal que* $L(G') = L(G)$.

Teorema 2 (Theorem 8.16 Aho Ullman vol 2). *Los lenguajes libres de contexto reconocibles por autómatas de pila determinísticos coinciden con los lenguajes* $LR(1)$.

Notar que una de las implicaciones resulta de que el algoritmo de parsing $LR(1)$ se implementa en un autómata de pila determinístico.

Parsing "bottom up"

La técnica de parsing determinista "bottom up" que opera linealmente consiste hacer sucesivas reducciones que nos lleven desde la cadena de entrada hasta el símbolo Start S'. Como resultado obtenemos, en orden invertido, la sucesión de derivaciones más a la derecha que producen la cadena dada.

Definición (Reducción). *Sea* $G = (N, T, P, S)$ *una gramática libre de contexto y supongamos la siguiente derivación a derecha*

$$S \underset{R}{\overset{*}{\Rightarrow}} \alpha Aw \underset{R}{\Rightarrow} \alpha\beta w \underset{R}{\overset{*}{\Rightarrow}} xw$$

Decimos que la forma sentencial $\alpha\beta w$ *puede ser* reducida *usando la producción* $A \to \beta$ *a la forma sentencial* αAw.

Pivote

Definición (Pivote o "handle"). *Un* pivote *de una forma sentencial* γ *es una pareja formada por una producción* $A \to \beta$ *y una posición en la cadena* γ *donde ocurre* β.

La reducción reemplaza la ocurrencia de β por A para producir una forma sentencia anterior en una derivación más a la derecha de γ.

Si

$$S' \underset{R}{\Rightarrow} \alpha Aw \underset{R}{\Rightarrow} \alpha\beta w$$

el pivote es $A \to \beta$ y la posición es $|\alpha| + 1$.

Importante: la expresion w a la derecha de β es una secuencia de terminales.

La técnica de parsing determinista "bottom up" que opera linealmente consiste en identificar unívocamente el pivote y hacer una reducción.

Ejemplo derivación, reducción y pivote

$$
\begin{aligned}
S' &\to S \\
S &\to aABe \\
A &\to Abc|b \\
B &\to d
\end{aligned}
$$

Sea $w = abbcde$

$$S' \underset{R}{\Rightarrow} \underline{S} \underset{R}{\Rightarrow} \underline{aABe} \underset{R}{\Rightarrow} aA\underline{d}e \underset{R}{\Rightarrow} a\underline{Abc}de \underset{R}{\Rightarrow} a\underline{b}bcde$$

Para cada forma sentencial el subrayado indica el pivote.

Ejemplo de parsing $LR(1)$

$$
\begin{aligned}
S' &\to S \\
S &\to aABe \\
A &\to Abc|b \\
B &\to d
\end{aligned}
$$

$$S' \underset{R}{\Rightarrow} S \underset{R}{\Rightarrow} aABe \underset{R}{\Rightarrow} aAde \underset{R}{\Rightarrow} aAcde \underset{R}{\Rightarrow} abbcde$$

PILA	INPUT	ACCION	OUTPUT
\$	$abbcde$\$	Desplazar a	
\$$a$	$bbcde$\$	Desplazar b	
\$$ab$	$bcde$\$	Reducir $A \to b$	$A \to b$
\$$aA$	$bcde$\$	Desplazar b	
\$$aAb$	cde\$	Desplazar c	
\$$aAbc$	de\$	Reducir $A \to Abc$	$A \to Abc$
\$$aA$	de\$	Desplazar d	
\$$aAd$	e\$	Reducir $B \to d$	$B \to d$
\$$aAB$	e\$	Desplazar e	
\$$aABe$	\$	Reducir $S \to aABe$	$S \to aABe$
\$$S$	\$	Reducir $S' \to S$	$S' \to S$
\$$S'$	\$	Aceptar	

Prefijo viable

Definition 1 (Prefijo viable)**.** Sea $G = \langle N, T, P, S \rangle$ una gramática independiente del contexto y supongamos que

$$S \overset{*}{\underset{R}{\Rightarrow}} \alpha A w \underset{R}{\Rightarrow} \alpha\beta w,$$

todos prefijos de la cadena $\alpha\beta$ son *prefijos viables* de la gramática G.

En general, un prefijo de una forma sentencial es viable si no continúa más allá del extremo derecho del pivote. Los prefijos viables son las cadenas que quedan almacenadas en la pila.

El conjunto de prefijos viables es un conjunto regular

Teorema 3. *El conjunto de prefijos viables de una gramática $LR(k)$ es regular.*

Items $LR(k)$

Un item de una gramática $LR(k)$ es una producción, un '.' en la parte derecha de la producción y una cadena de terminales de longitud menor o igual que k.

Definition 2 (Item válido $LR(k)$)**.** Fijemos una gramática $G = (N, T, P, S)$ libre de contexto. Supongamos $A \to \alpha\beta \in P$. Un item $[A \to \alpha.\beta, u]$, con $u \in T^*$ y $|u| \leq k$, es un *item* $LR(k)$ *válido* para el prefijo viable $\eta\alpha$ si existe una derivación a derecha tal que

$$S \overset{*}{\underset{R}{\Rightarrow}} \eta A w \underset{R}{\Rightarrow} \eta\alpha\beta w \quad \text{ y } u \in \text{PRIMEROS}_k(w).$$

Agregaremos además items con $u = \$$ y pediremos pedimos que $w = \lambda$. Sirve para indicarle al parser el fin de la cadena de entrada.

En particular, para $k = 0$, un item $LR(0)$ $[A \to \alpha.\beta]$ es válido para el prefijo viable $\eta\alpha$ si existe una derivación a derecha tal que

$$S \overset{*}{\underset{R}{\Rightarrow}} \eta A w \underset{R}{\Rightarrow} \eta\alpha\beta w$$

Ejemplo de items $LR(1)$

Consideremos esta gramática

$$\begin{array}{rcl} S' & \to & S \\ S & \to & BB \\ B & \to & aB|b \end{array}$$

De la derivación $S \overset{*}{\underset{R}{\Rightarrow}} aaBab \underset{R}{\Rightarrow} aaaBab$ concluímos que el item $[B \to a.B, a]$ es válido para el prefijo viable aaa.

De la derivación $S \overset{*}{\underset{R}{\Rightarrow}} BaB \underset{R}{\Rightarrow} BaaB$ concluímos que el item $[B \to a.B, \$]$ es válido para el prefijo viable Baa.

De la derivación $S' \overset{*}{\underset{R}{\Rightarrow}} S' \underset{R}{\Rightarrow} S$ concluímos que el item $[S' \to .S, \$]$ es válido para el prefijo viable λ.

El conjunto de prefijos viables $LR(1)$ **es un lenguaje regular**

Dada $G = (N, T, P, S)$, gramática $LR(1)$ definimos el AFND-λ $M = \langle Q, N \cup T, \delta, q_0, Q \rangle$ donde

- Q es el conjunto de items válidos $LR(1)$ de la gramática G
- $q_0 = [S' \to .S, \$]$
- $\delta : Q \times (T \cup N \cup \{\lambda\}) \to$ subconjuntos de Q se define así:

$$[A \to \alpha.X\beta, a] \xrightarrow{X} [A \to \alpha X.\beta, a]$$
$$[A \to \alpha.B\beta, a] \xrightarrow{\lambda} [B \to .\gamma, b]$$

para $X \in (N \cup T)$, $a \in T \cup \{\$\}$, $b \in \text{PRIMEROS}(\beta a)$.

Recordemos la noción de clausura-λ

Sea $\langle Q, \Sigma, \delta, q_0, F\rangle$ un AFND-λ.
La función de transición $\delta : Q \times (\Sigma \cup \{\lambda\}) \to$ subconjuntos de Q.

Definimos $R \subseteq Q \times Q$, $(q, p) \in R$ si y solo si $p \in \delta(q, \lambda)$.
Escribimos R^* para la clausura reflexo-transitiva de R.
Notemos que $(p, q) \in R^*$ si hay un camino de transiciones-λ de p a q.

Definimos, la Clausura-λ de un estado $p \in Q$

$$Cl_\lambda(p) = \{q : (p, q) \in R^*\}$$

La Clausura-λ de un conjunto de estados $P \subseteq Q$

$$Cl_\lambda(P) = \bigcup_{p \in P} Cl_\lambda(p)$$

Y definimos la función de transición $\hat{\delta} : Q \times \Sigma^* \to$ subconjuntos de Q,

$$\hat{\delta}(q, \lambda) = Cl_\lambda(q)$$
$$\hat{\delta}(q, xa) = Cl_\lambda\Big(\bigcup_{r \in \hat{\delta}(q,x)} \delta(r, a)\Big)$$

El conjunto de prefijos viables $LR(1)$ es un lenguaje regular

Sea $G = (N, T, P, S)$ gramática $LR(1)$
Sea AFND-λ $M = \langle Q, N \cup T, \delta, q_0, Q\rangle$ asociago a G.

Dado que G es $LR(1)$ en cada paso hay una única derivación a la derecha con un look-ahead de 1 símbolo, por lo tanto, hay AFD $\tilde{M} = (\tilde{Q}, N \cup T, \tilde{\delta}, \tilde{q}_0, \tilde{Q}\rangle$, donde

$$\tilde{Q} = \{Cl_\lambda(q) : q \in Q \text{ y existe } \alpha \in (N \cup T)^*, q \in \hat{\delta}(q_0, \alpha)\}$$
$$\tilde{q}_0 = Cl_\lambda(q_0)$$
$$\tilde{\delta} :\ \tilde{Q} \times (N \cup T) \to \tilde{Q},$$
$$\tilde{\delta}(q, X) = Cl_\lambda\Big(\{[A \to \alpha X.\beta, a] : [A \to \alpha.X\beta, a] \in q\}\Big), \text{ para} X \in (N \cup T)$$

El conjunto de prefijos viables $LR(1)$ es un lenguaje regular

De la definición de AFD $\tilde{M}$ obtenemos

$[A \to \alpha.\beta, a] \in \tilde{\delta}(q_0, \gamma)$ si y solo si
$[A \to \alpha.\beta, a]$ es un item válido para el prefijo viable γ.

Es decir,

$[A \to \alpha.\beta, a] \in \tilde{\delta}(q_0, \gamma)$ si y solo si
$S' \underset{R}{\overset{*}{\Rightarrow}} \eta A w \underset{R}{\Rightarrow} \gamma\beta w$, $\gamma = \eta\alpha$ y PRIMEROS$(w) = a$ o ($w = \lambda$ y $a = \$$).
Dado que conjunto de estado finales de $\tilde{M}$ es todo el conjunto $\tilde{Q}$,

$$L(\tilde{M}) = \{\gamma \in (N \cup T)^* : \gamma \text{ es prefijo viable de } G\}$$

□

Algoritmo de parsing $LR(1)$

Sea $G = (N, T, P, S)$ gramática $LR(1)$.

Sea AFD $\tilde{M} = \langle \tilde{Q}, (N \cup T), \tilde{\delta}, q_0, \tilde{Q} \rangle$ tal que $L(\tilde{M})$ = prefijos viables de G.

Input	$a_1 a_2 \dots a_n \$$
Output	Sucesión de producciones de la derivación más a la derecha, de la cadena dada en Input, en orden inverso.
Pila	$q_0 X_1 q_1 \dots X_m q_m$, donde los $q \in \tilde{Q}$ y $X \in (N \cup T)$
Tabla	$ACCION : \tilde{Q} \times (T \cup \{\$\}) \to$ {Desplazar q,Reducir $A \to \beta$,Aceptar, Error } (a continuación damos la definición)
Función	$IR : \tilde{Q} \times (T \cup N) \to \tilde{Q}$ es la función de transición $\tilde{\delta}(p, X)$

$IR(q, A)$ para $A \in N$ se usa en algoritmo al apilar un estado $IR(q, a)$ para $a \in T \cup \{\$\}$ se usa para definir $ACCION$

Tabla $ACCION$

Sea $G = (N, T, P, S)$ gramática $LR(1)$.

Sea AFD $\tilde{M} = \langle \tilde{Q}, (N \cup T), \tilde{\delta}, q_0, \tilde{Q} \rangle$ tal que $L(\tilde{M})$ = prefijos viables de G.

$ACCION : \tilde{Q} \times (T \cup \{\$\}) \to$ {Desplazar q,Reducir $A \to \beta$,Aceptar, Error }

Si $[A \to \alpha.a\beta, b]$ en q_i con $a \in T$ y $IR(q_i, a) = q_j$ entonces
$ACCION(q_i, a)$= Desplazar q_j

Si $[A \to \alpha., a]$ en q_i y $A \neq S'$ con $a \in T \cup \{\$\}$ entonces
$ACCION(q_i, a)$= Reducir $A \to \alpha$

Si $[A \to S., \$]$ en q_i entonces
$ACCION(q_i, \$)$= Aceptar

Sino $ACCION(q_i, \$)$= Error

Algoritmo parsing $LR(1)$

Input $a_1 \dots a_n \$$

Puntero en posición 1 de cadena de entrada

Pila: q_0

Repetir

Sea q el estado en el tope de la pila

Sea a el símbolo actual del input

Si $ACCION(q, a)$=Desplazar p entonces

Apilar a

Apilar p

Avanzar una posición el puntero del input

Si $ACCION(q, a)$= Reducir $A \to \beta$ entonces

Desapilar $2|\beta|$ símbolos de la pila

Sea p el tope de la pila

Apilar A

Apilar $IR(p, A)$

Output $A \to \beta$

hasta ($ACCION(q, a)$= Aceptar o $ACCION(q, a)$=Error)

Algoritmo LR tiene complejidad lineal

Teorema 4 (Teorema 5.13 Aho Ullman vol 1)**.** *El algoritmo de parsing $LR(k)$ realiza una cantidad de operaciones lineal en el tamaño de la entrada.*

¿Los autómatas finitos se cuelgan?

Para demostrar el teorema debemos remontarnos al cómputo en los autómatas de pila.

Veamos primero el caso en autómatas finitos.

Un autómata finito ¿se cuelga?

No, la cantidad de transiciones que realiza un AFD o un AFND es exactamente la cantidad de símbolos de la entrada. En caso de tratarse de un AFND-λ, aquí hay transiciones λ que no permiten acotar la cantidad de transiciones. Sin embargo, es posible construir un AF equivalente que resulta de remover los ciclos de transiciones-λ.

En tal AFD la cantidad de transiciones está acotada por la cantidad de estados multiplicado por la longitud de la entrada.

Autómatas de pila determinísticos

Recordemos

Definición (Autómata de Pila determinístico, página 184 Aho Vol1)**.** *Un autómata de pila $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ es determinístico si para cada $a \in \Sigma$, $q \in Q$ y $Z \in \Gamma$ se cumple que*

$\delta(q, a, Z)$ contiene a lo sumo un elemento y $\delta(q, \lambda, Z) = \emptyset$

$\delta(q, a, Z) = \emptyset$ y $\delta(q, \lambda, Z)$ tiene a lo sumo un elemento.

La cantidad de transisiones que realiza un AP determinístico no está acotada por el tamaño de la entrada. La ejecución depende no solamente de la entrada sino también del contenido de la pila, cuyo tamaño no está acotado.

Es posible que un autómata de pila determinístico realice una cantidad infinita de λ-movimienos desde alguna configuración sin leer ningún símbolo de la entrada. Decimos que estas configuarciones ciclan.

¿Los autómatas de pila determinísticos se cuelgan?

Teorema 5 (Teorema 2.2 Aho Ullman vol 2)**.** *Para todo autómata de pila determinístico $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ hay otro P' tal que $L(P) = L(P')$ y P' no tiene configuraciones que ciclen.*

La demostración se basa en el Lema 2.20 de Aho y Ullman vol 2.

La función de transición es $\delta : Q \times \Sigma \times \Gamma \to Q \times \Gamma^*$. Supongamos que en cada transición de P se escriben en la pila menos que ℓ símbolos de Γ. Entonces la cantidad de transiciones sin ciclar es a lo sumo

$$Q \times (\Gamma^\ell)^{Q\times\Gamma}$$

A partir de esta cantidad es posible definir un autómata determinístico que verifica que no se pasa dos veces por la misma configuración. □

Notar que la cota calculada es una constante.

Demostración algoritmo LR complejidad lineal

Supongamos la entrada es la cadena $a_1 \dots a_n\$$.

El algoritmo en cada iteración realiza acción Desplazar o acción Reducir

Definimos una C-configuración $(q_0 x_1 q_1 \dots q_m x_m, a_i a_{i+1} \dots a_n\$)$

- incial $(q_0, a_1 \dots a_n\$)$.
- despues de Desplazar

- despues de Reducir, en caso de que la pila se haya achicado
Asignamos a cada C-configuración

$$\text{valor} = \#\text{ símbolos de la pila} + 2\ \#\text{ símbolos por leer}$$

Entonces C-configuración inicial tiene valor$= 0 + 2n = 2n$.
Si C_1 y C_2 son C-configuraciones sucesivas

- C_2 surge de Desplazar (y leer 1 símbolo de la entrada) y
  su valor es 2 menos que el de C_1; o bien,
- C_2 surge de Reducir y su valor es 1 menos que C_1, o menor.

Luego, el parsing de una cadena de longitud n pasa a lo sumo por
$2n$ C-configuraciones. Necesitamos ver que entre dos C-configuraciones hay una cantidad constante de operaciones.

Para eso simulamos el algoritmo $LR(k)$ en APD, donde la pila de autómata coincide con la pila de algoritmo.

Por el Teorema 2.22 de Aho Ullman vol 1, si un APD no reduce su pila nunca más y no lee más la entrada, está en un ciclo. La única posibilidad de que el algoritmo no reduzca la pila es que haya una derivación más a la derecha arbitrariamente larga. Pero esto es imposible, ya que las gramáticas LR no tienen ciclos ya que no son ambiguas, entonces cada cadena del lenguaje aceptado tiene una única derivación más a la derecha.

La máxima cantidad de operaciones de un APD sin ciclar está acotada por la constante $|Q| \times (|\Gamma|^{\ell})^{|Q|\times|\Gamma|}$ (Lemma 2.20 Aho Ullman vol 1). Conlcuimos que el APD no entra en un ciclo y por lo tanto la cantidad total de pasos de la derivación para aceptar la cadena de entrada es lineal en la longitud de la cadena. □