Teoría de Lenguajes
Teórica 12: Gramáticas LR y Parsing $LR(1)$

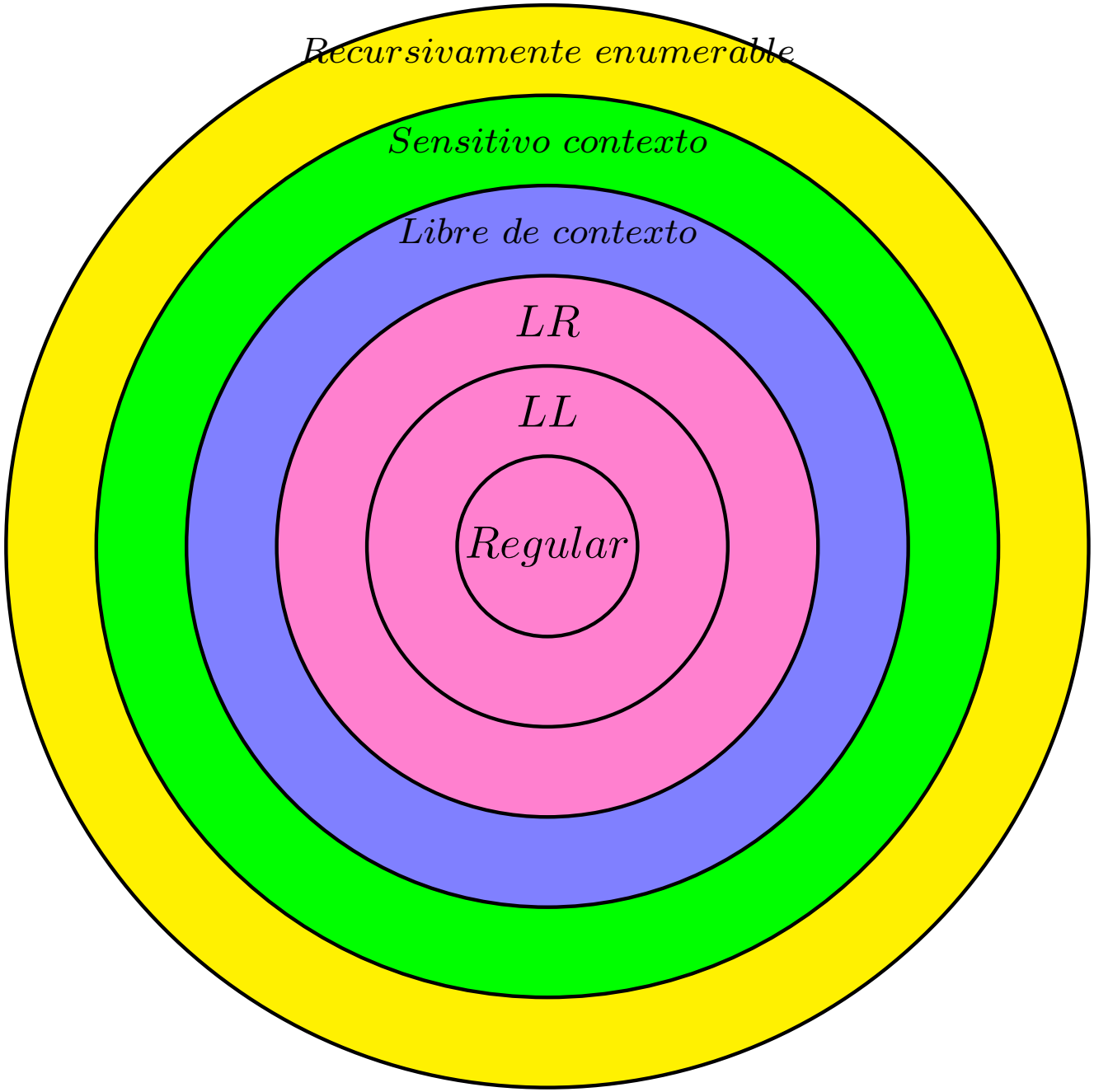
Primer Cuatrimestre 2024

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 2, 5.1 y 5.2.

A. V. Aho, Sethi, J. D. Ullman, Segunda edicion , 2006. (el libro del dragón)
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 $LR(k)$

Las gramáticas $LR(k)$, con k un número entero mayor o igual que 0, son gramáticas libres de contexto no ambiguas para las cuales dada una expresión del lenguaje se puede encontrar su derivación más a la derecha de manera "bottom-up", de modo tal que en cada paso de la derivación está determinada por los símbolos ya leídos de la cadena de entrada y k símbolos más. De esta forma, cada paso de la derivación se resuelve esencialmente en tiempo constante.

La definición de gramática $LR(k)$se debe a Knuth (1965). Luego la técnica fue mejorada por muchos otros entre ellos De Remer (1969), Karenjack (196), Aho Ullman (1971).
El concepto $LR(k)$ se extendio a gramáticas sensitivas al contexto Walters (1970).

Los lenguajes $LR(k)$ son exactamete los lenguajes reconocidos por autómatas de pila determinísticos.

Gramáticas $LR(k)$

Sea $G = (T, N, P, S)$ libre de contexto y no ambigua. Consideremos la gramática extendida $G' = (T, N \cup \{S'\}, P \cup \{S' \to S\}, S')$.
Sea w una cadena de $L(G)$. Dado que G es no ambigua hay una única secuencia de formas sentenciales $\alpha_0, \alpha_1, \ldots \alpha_m$ tal que

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

donde S' es un nuevo símbolo de inicio y las derivaciones en G'.

El parsing a derecha de w es la secuencia de las m producciones usadas en la derivación de w. Las gramáticas $LR(k)$ aseguran que α_i es determinable teniendo en cuenta los k símbolos más del input w que los ya leídos.

Gramáticas $LR(k)$

Definición (Gramática $LR(k)$)

Dada una gramática libre de contexto $G = (N, T, P, S)$ *definimos la gramática aumentada* $G' = (N \cup \{S'\}, T, P \cup \{S' \to S\}, S')$.
La gramática G *es* $LR(k)$, *con* $k \geq 0$, *si estas tres condiciones*

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

$$S' \underset{R}{\overset{*}{\Rightarrow}} \gamma B z.... \underset{R}{\Rightarrow} \alpha\beta z...$$

$$|z| = k$$

ó

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

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

$$|z| < k$$

implican $\alpha = \gamma$, $A = B$.

Ejemplo: gramática no $LR(0)$, sí $LR(1)$

$$\begin{aligned} S' &\to S \\ S &\to Sa \mid a \end{aligned}$$

Este par de derivaciones muetran que G no es $LR(0)$

$$S' \underset{R}{\overset{*}{\Rightarrow}} S' \underset{R}{\Rightarrow} S$$

$$S' \underset{R}{\overset{*}{\Rightarrow}} S \underset{R}{\Rightarrow} Sa$$

Para decidir si la gramática G cumple la definición de $LR(1)$ debemos considerar todo par de derivaciones de uno o más pasos. Hay 3 casos.

Primer Caso. Supongamos $\ell > 0$ Consideremos este par de derivaciones

$$S' \underset{R}{\overset{*}{\Rightarrow}} S \quad \underset{R}{\Rightarrow} a$$
$$S' \underset{R}{\overset{*}{\Rightarrow}} Sa^{\ell} \underset{R}{\Rightarrow} aa^{\ell}$$

La única asignación posible para considerar la definición $LR(1)$ es:

$$\alpha = \gamma = \lambda, \quad A = B = S, \quad \beta = a,$$

No hay z de longitud 1 coincidente, por lo tanto no hay que verificar la condición. Concluimos que este par de derivaciones es consistente con la definición de $LR(1)$.

Segundo Caso. Consideremos este par de derivaciones, para $j > 0$, $\ell > 0$,

$$S' \underset{R}{\overset{*}{\Rightarrow}} Sa^j \underset{R}{\Rightarrow} aa^j$$
$$S' \underset{R}{\overset{*}{\Rightarrow}} Sa^{j+\ell} \underset{R}{\Rightarrow} aa^{j+\ell}$$

Se cumple la condición de $LR(1)$ para $z = a$,

$$\alpha = \gamma = \lambda, \quad A = B = S, \quad \beta = a,$$

Concluimos que este par de derivaciones es consistente con la definición de $LR(1)$.

Tercer Caso. Consideremos este par de derivaciones,

$$S' \underset{R}{\overset{*}{\Rightarrow}} S' \underset{R}{\Rightarrow} S$$
$$S' \underset{R}{\overset{*}{\Rightarrow}} S \underset{R}{\Rightarrow} Sa$$

No hay z de longitud 1 coincidente, por lo tanto no hay que verificar la condición.
Concluimos que este par de derivaciones es consistente con la definición de $LR(1)$.

De los tres casos concluímos que la gramática G es $LR(1)$.

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

$$\begin{aligned} S &\rightarrow Ab|\ Bc \\ A &\rightarrow Aa|\ \lambda \\ B &\rightarrow Ba|\ \lambda \end{aligned}$$

Observemos que para todo $k \geq 0$ tenemos

$$S' \underset{R}{\overset{*}{\Rightarrow}} Aa^k b \underset{R}{\Rightarrow} a^k b$$
$$S' \underset{R}{\overset{*}{\Rightarrow}} Ba^k c \underset{R}{\Rightarrow} a^k c$$

Tenemos que para $z = a^k$ y $\beta = \lambda$ no se cumple la condición porque $A \neq B$.

Notar que G no es $LL(k)$ para ningun k porque es recursiva a izquierda.
Sin embargo $L(G) = a^*b|a^*c$ es regular por lo tanto admite una gramática LL y también admite una gramática LR.

Las gramáticas LR son no ambiguas

Teorema

Toda gramática LR es no ambigua.

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

$$S \underset{R}{\Rightarrow} \alpha_1 \underset{R}{\Rightarrow} \alpha_2 \ldots \underset{R}{\Rightarrow} \alpha_n \underset{R}{\Rightarrow} w$$
$$S \underset{R}{\Rightarrow} \beta_1 \underset{R}{\Rightarrow} \beta_2 \ldots \underset{R}{\Rightarrow} \beta_m \underset{R}{\Rightarrow} w$$

Consideremos el menor i tal que $\alpha_{n-i} \neq \beta_{m-i}$. Sea $y = \alpha_{n-(i-1)}$.

$$S \overset{*}{\underset{R}{\Rightarrow}} \alpha_{n-i} \underset{R}{\Rightarrow} y$$
$$S \overset{*}{\underset{R}{\Rightarrow}} \beta_{m-i} \underset{R}{\Rightarrow} y$$

Sea $z \in T^*$ el sufijo más largo de α_{n-i} (solamente símbolos terminales) que también es sufijo de y (puede ser vacío). Dado que $\alpha_{n-i} \neq \beta_{m-i}$ es imposible que la gramática sea $LR(k)$. $\square$

Lenguajes LR

Teorema (Theorem 8.16 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 (Theorem 8.10 junto con 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.

Teorema (Theorem 8.1 Aho Ulman vol 2)

Toda gramatica $LL(k)$ sin producciones inutiles es $LR(k)$.

Ejemplo derivación a derecha, 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*, concepto que definiremos a continuación.

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

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

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

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

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

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 de parsing $LR(1)$

$$S' \to S$$
$$S \to aABe$$
$$A \to Abc|b$$
$$B \to d$$

Sea INPUT $abbcde$. $S' \underset{R}{\Rightarrow} \underline{S} \underset{R}{\Rightarrow} \underline{aABe} \underset{R}{\Rightarrow} aA\underline{de} \underset{R}{\Rightarrow} a\underline{Abcde} \underset{R}{\Rightarrow} a\underline{b}bcde$

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

Prefijo viable

Definición (Prefijo viable)

Sea $G = \langle N, T, P, S \rangle$ una gramática independiente del contexto y supongamos que

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

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

Un prefijo de una forma sentencial es viable si NO sobrepasa el extremo derecho del pivote. Los prefijos viables son las cadenas que quedan almacenadas en la pila durante el parsing.

El conjunto de prefijos viables es un conjunto regular

Teorema

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.

$$\begin{aligned} S' &\rightarrow S \\ S &\rightarrow BB \\ B &\rightarrow aB|b \end{aligned}$$

Ejemplos
$[S' \rightarrow .S, \$]$
$[B \rightarrow a.B, a]$
$[B \rightarrow aB., \$]$

Items $LR(k)$

Definición (Item válido $LR(k)$ para prefijo viable)

Fijemos una gramática $G = (N, T, P, S)$ libre de contexto.
Sea $A \to \alpha\beta \in P$. 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

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

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

Por otro lado, 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

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

Ejemplo de items $LR(1)$

Consideremos esta gramática

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

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

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

De la derivación $S' \underset{R}{\overset{*}{\Rightarrow}} S' \underset{R}{\Rightarrow} S$ 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

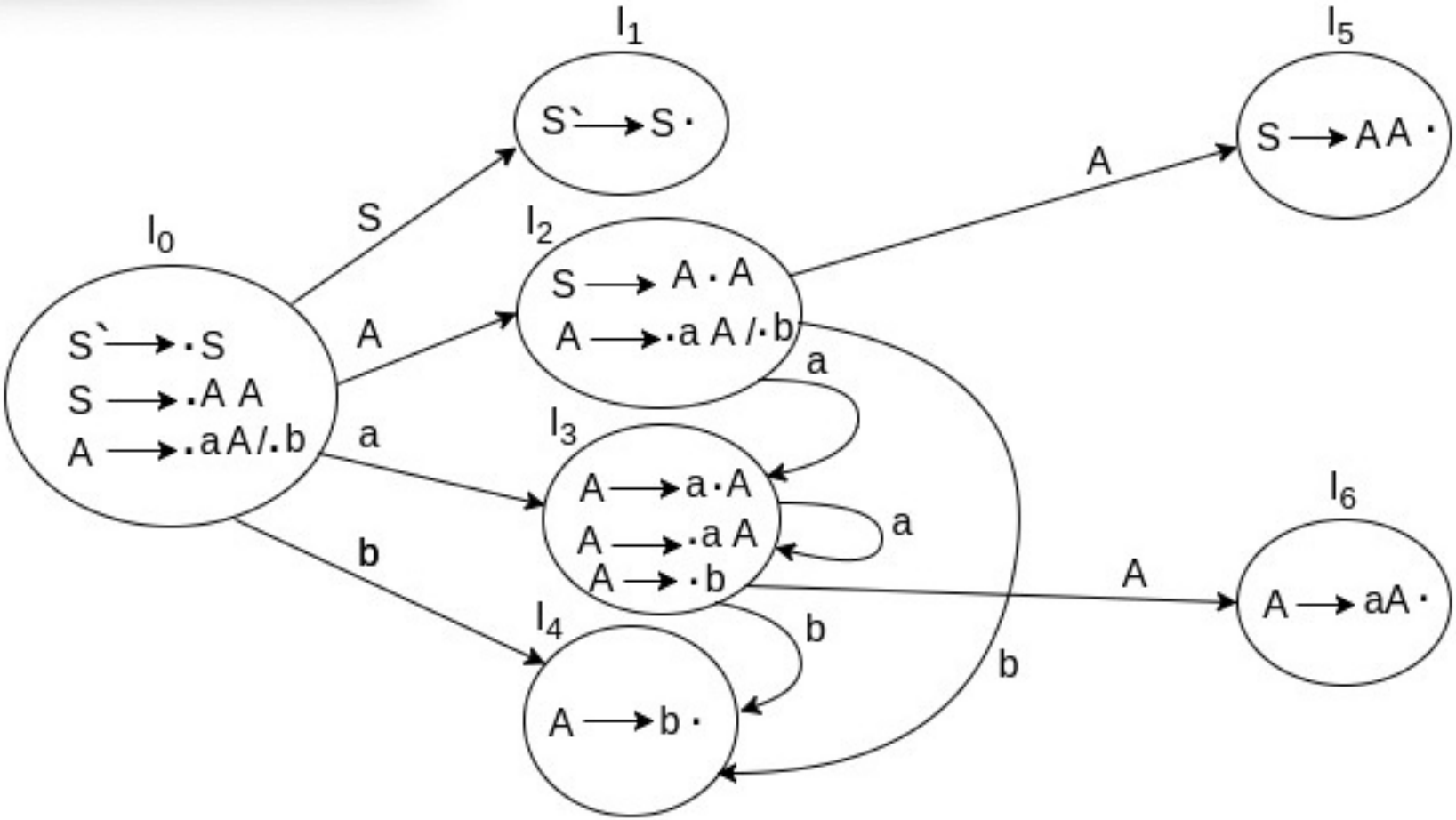
- Q es el conjunto de items válidos $LR(1)$ de la gramática G
- $q_0 = [S' \to .S, \$]$
- $\delta : Q \times (T \cup N \cup \{\lambda\}) \to$ subconjuntos de Q se define así:

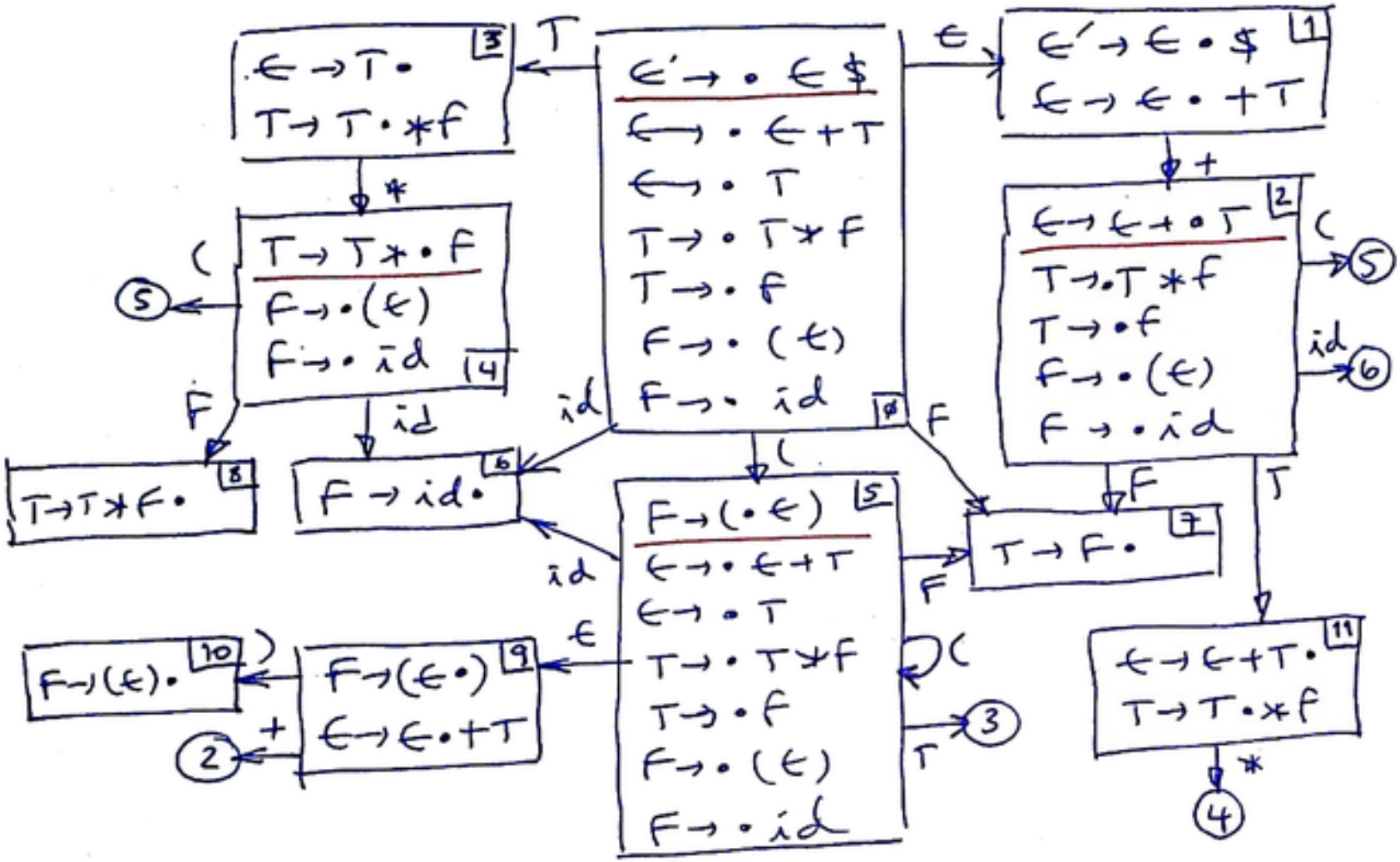
$$[A \to \alpha.X\beta, a] \xrightarrow{X} [A \to \alpha X.\beta, a]$$

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

para $a \in T \cup \{\$\}$, $A, B \in N$, $X \in (N \cup T)$, $b \in$ PRIMEROS(βa).

Pero queremos llegar a un AFD sin transiciones λ




E → T •
T → T • * f
3
T
E' → • E $
E → • E + T
E → • T
T → • T * f
T → • f
F → • (E)
F → • id
0
E
E' → E • $
E → E • + T
1
+
E → E + • T
T → • T * f
T → • f
F → • (E)
F → • id
2
(
5
id
6
T → T * • f
F → • (E)
F → • id
14
*
F
id
T → T * F •
8
F → id •
6
F → (• E)
E → • E + T
E → • T
T → • T * f
T → • f
F → • (E)
F → • id
5
T → F •
7
F
T
E → E + T •
T → T • * f
11
*
4
F → (E) •
10
F → (E •)
E → E • + T
9
+
2
3

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

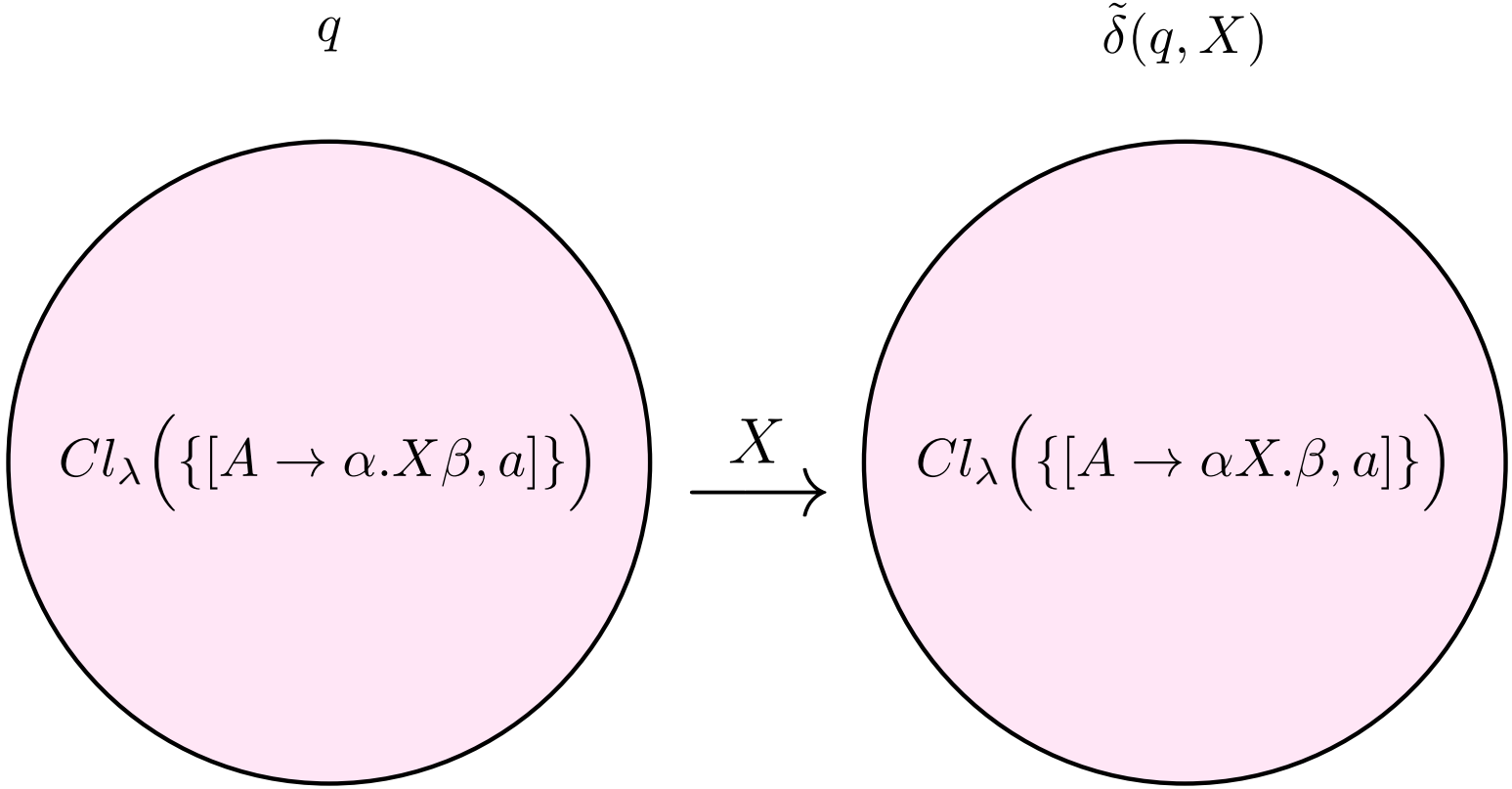
Sea $G = (N, T, P, S)$ gramática $LR(1)$
Sea AFND-λ $M = \langle Q, N \cup T, \delta, q_0, Q \rangle$ asociado a G.

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

$$\tilde{Q} = \{Cl_\lambda(q) : q \in Q \text{ y existe } \alpha \in (N \cup T)^*, q \in \hat{\delta}(q_0, \alpha)\}$$
$$\tilde{q}_0 = Cl_\lambda(q_0) = Cl_\lambda([S' \to .S, \$])$$

$$\tilde{\delta}: \quad \tilde{Q} \times (N \cup T) \to \tilde{Q},$$
$$\tilde{\delta}(q, X) = \bigcup_{[A \to \alpha.X\beta, a] \in q} Cl_\lambda\Big([A \to \alpha X.\beta, a]\Big), \text{ para} X \in (N \cup T)$$

La función de transición $\tilde{\delta} : \tilde{Q} \times (N \cup T) \to \tilde{Q}$

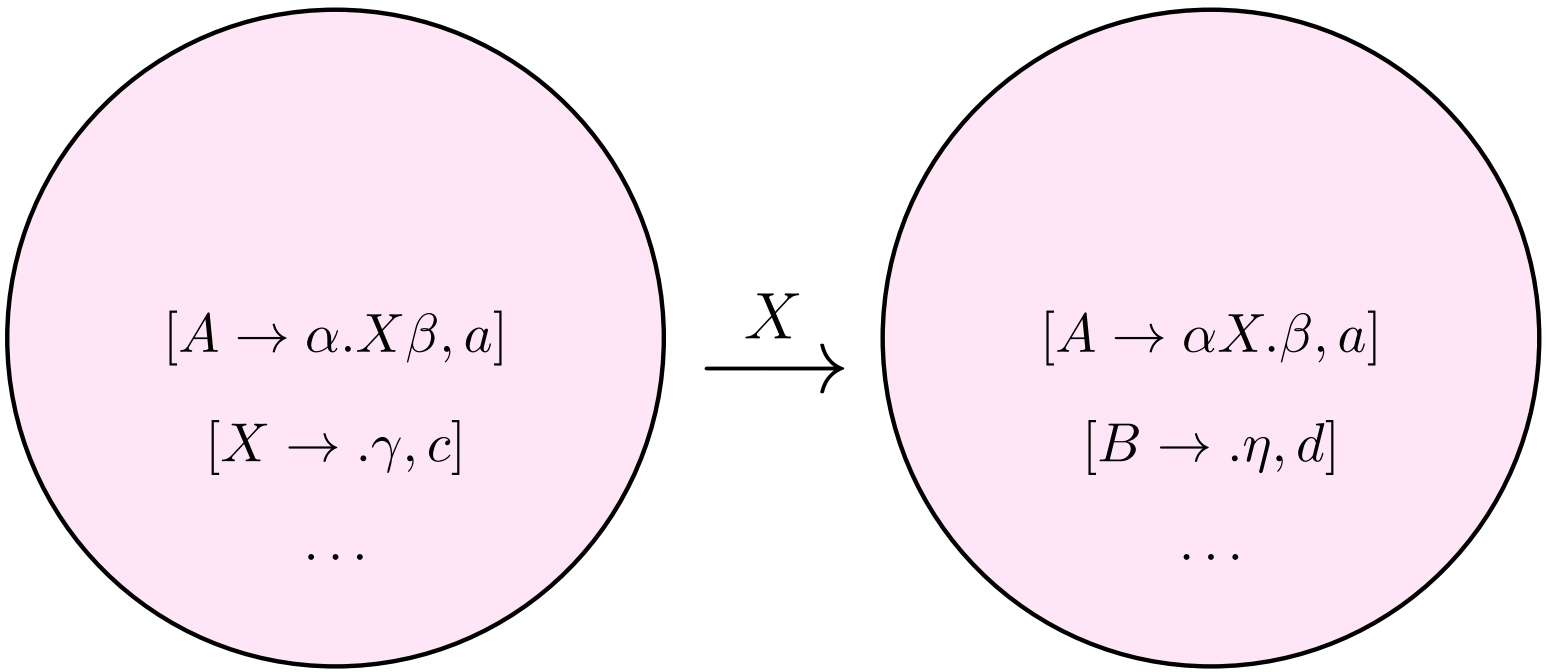


donde $X \in (N \cup T)$

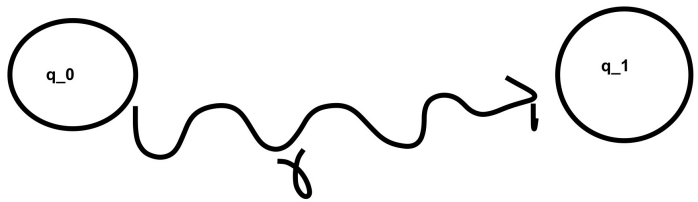
La función de transición $\tilde{\delta} : \tilde{Q} \times (N \cup T) \to \tilde{Q}$

En caso de que $X \in N$ el estado q tiene al menos dos items:

$$q = Cl_\lambda\Big(\{[A \to \alpha.X\beta, a]\}\Big) \qquad \tilde{\delta}(q, X) = Cl_\lambda\Big(\{[A \to \alpha X.\beta, a]\}\Big)$$



donde $c \in$ PRIMEROS(βa) si $\beta = B\varphi$ y $d \in$ PRIMEROS(φa)



$\gamma = \eta\alpha$

$q_1 \in C(([A \to \alpha.\beta, a]$

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

a[1] en PRIMEROS(..)

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

Dado que AFD $\tilde{M} = (\tilde{Q}, (N \cup T), \tilde{\delta}, \tilde{q}_0, \tilde{Q})$, el conjunto de estado finales de $\tilde{M}$ es todo el conjunto $\tilde{Q}$ y por lo tanto

$$L(\tilde{M}) = \{\gamma \in (N \cup T)^* : \tilde{\delta}(\tilde{q}_0, \gamma) \in \tilde{Q}\}$$

Dado que $q_0 = [S' \to .S, \$]$,

$[A \to \alpha.\beta, a] \in \tilde{\delta}(q_0, \gamma)$ si y solo si existe $\eta \in (N \cup T)^*$ y $w \in T^*$ tal que

- $\gamma = \eta\alpha$
- $S' \underset{R}{\Rightarrow} S \overset{*}{\underset{R}{\Rightarrow}} \eta A w \underset{R}{\Rightarrow} \eta\alpha\beta w$
- PRIMEROS$(w) = a$. Notar que puede ser $w = \lambda$ y $a = \$$.

Luego,

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

□

Algoritmo de parsing $LR(1)$

Sea $G = (N, T, P, S)$ gramática $LR(1)$.
Sea AFD $\tilde{M} = \langle \tilde{Q}, (N \cup T), \tilde{\delta}, \tilde{q}_0, \tilde{Q} \rangle$ 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	$ACCION : \tilde{Q} \times (T \cup \{\$\}) \to$ {apilar 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 $\tilde{M} = \langle \tilde{Q}, (N \cup T), \tilde{\delta}, \tilde{q}_0, \tilde{Q} \rangle$ tal que $L(\tilde{M})$ = prefijos viables de G.

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

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

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

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

Sino $ACCION(q_i, a)$= Error

Ejemplo de parsing $LR(1)$ con estados en la pila

$$S' \to S \quad S \to aABe \quad A \to Abc|b \quad B \to d$$

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

PILA	INPUT	ACCION	OUTPUT
q_0	$abbcde\$$	ACCION(q_0, a)= Apilar $a\ q_1$	
q_0aq_1	$bbcde\$$	ACCION(q_1, b)= Apilar $b\ q_2$	
$q_0aq_1bq_2$	$bcde\$$	ACCION(q_2, b)=Reducir $A \to b$	$A \to b$
		Desapilar q_2b Apilar A IR(q_1, A)	
$q_0aq_1Aq_3$	$bcde\$$	ACCION(q_3, b)= Apilar $b\ \ q_4$	
$q_0aq_1Aq_3bq_4$	$cde\$$	ACCION(q_4, c)=Apilar $c\ \ q_5$	
$q_0aq_1Aq_3bq_4cq_5$	$de\$$	ACCION(q_5, d)=Reducir $A \to Abc$	$A \to Abc$
		Desapilar $q_5cq_4bq_3A$, Apilar A IR(q_1, A)	
$q_0aq_1Aq_6$	$de\$$	ACCION(q_6, d)= Apilar $d\ q_7$	
$q_0aq_0q_1Aq_6dq_7$	$e\$$	ACCION(q_7, e)=Reducir $B \to d$	$B \to d$
		Desapilar q_7d, Apilar B IR(q_6, B)	
$q_0aq_1Aq_6Bq_8$	$e\$$	ACCION(q_8, e)= Apilar $e\ q_9$	
$q_0aq_1Aq_6Bq_8eq_9$	$\$$	ACCION$(q_9, \$)$=Reducir $S \to aABe$	$S \to aABe$
		Desapilar $q_9eq_8Bq_6Aq_1a$,Apilar S IR(q_0, S)	
q_0Sq_{10}	$\$$	ACCION$(q_{10}, \$)$=Reducir $S' \to S$	$S' \to S$
		Desapilar $q_{10}S$, Apilar S' IR(q_{10}, S')	
$q_0S'q_{11}$	$\$$	ACCION$(q_{11}, \$)$= Aceptar	

Algoritmo parsing $LR(1)$

Input $a_1 \dots a_n\$$
Puntero en posición 1 de cadena de entrada

Pila: $\tilde{q}_0$
Repetir

Sea q el estado en el tope de la pila

Sea a el símbolo apuntado actualmente del input

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

Apilar a
Apilar p
Avanzar una posición el puntero del input

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

Desapilar $2|\alpha|$ símbolos de la pila
Sea p el tope de la pila
Apilar A
Apilar $IR(p, A)$
Output $A \to \alpha$

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

Algoritmo LR tiene complejidad lineal

Teorema (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.

Para demostrar el teorema necesitamos unos resultados sobre el cómputo en los autómatas de pila.

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 transiciones 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.

Es posible que un autómata de pila determinístico realice una cantidad infinita de λ-movimientos desde alguna configuración. Decimos que estas configuraciones ciclan.

Ciclos

Definición (Configuracion que ciclan, Aho Ullman vol 1 pagina 187)

Sea $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ *un autómata de pila determinístico. Una configuración* (q, w, α)*, con* $|\alpha| \geq 1$*, cicla si*

$$(q, w, \alpha) \vdash (p_1, w, \beta_1) \vdash (p_2, w, \beta_2) \vdash \ldots$$

con $|\beta_i| \geq |\alpha|$ *para todo* i*.*

Así, una configuración cicla si P realiza un número infinito de movimientos sin leer ningún símbolo de la entrada y el tamaño de la pila es de tamaño mayor of igual que $|\alpha|$. La pila puede crecer indefinidamente o ciclar entre distintas cadenas.

¿Los autómatas de pila determinísticos se cuelgan?

Teorema (Teorema 2.22 Aho Ullman vol 1, pagina 207)

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 de este Teorema usa el Lema 2.20 de Aho y Ullman vol 1 (pagina 172).

2.5.2. Variants of Pushdown Automata

In this section we shall define some variants of PDA's and relate the languages defined to the original PDA languages. First we would like to bring out a fundamental aspect of the behavior of a PDA which should be quite intuitive. This can be stated as "What transpires on top of the pushdown list is independent of what is under the top of the pushdown list."

LEMMA 2.20

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. If $(q, w, A) \vdash^n (q', e, e)$, then $(q, w, A\alpha) \vdash^n (q', e, \alpha)$ for all $A \in \Gamma$ and $\alpha \in \Gamma^*$.

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 pasa dos veces por la misma configuración. □

Demostración algoritmo LR complejidad lineal

Supongamos la entrada es la cadena $a_1 \dots a_n\$$.
El algoritmo en cada iteración realiza acción apilar o acción Reducir
Definimos una C-configuración $(q_0 x_1 q_1 \dots x_m q_m, a_i a_{i+1} \dots a_n\$)$
- incial $(q_0, a_1 \dots a_n\$)$.
- despues de apilar
- despues de Reducir, en caso de que la pila se haya achicado
Asignamos a cada C-configuración

valor= # símbolos de la pila + 3 # símbolos por leer

Entonces C-configuración inicial tiene valor= $1 + 3n$.
Si C_1 y C_2 son C-configuraciones sucesivas

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

Luego, dada un input de longitud n si el algoritmo termina, pasa a lo sumo por $3n + 1$ 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. Entoces el algoritmo de parsing también está en un ciclo.

Si la palabra no pertenece al lenguaje, sabemos que el algoritmo lo detecta. Dado que nuestra gramática es LR, para cada palabra que pertenece al lenguaje, hay una única derivación más a la derecha. Entoces para que el algoritmo haya entrado en un ciclo si no hay una palabra con infiitas derivaciones más a la derecha arbitrariamente largas. Pero esto es imposuble porque las gramática LR no son ambiguas.
Concluimos que el algoritmo no cicla. Es decir, no pasa por el mismo estado y tope de pila sin haber leído nuevos símbolos de entrada.

Dado que la función de transición de APD es $\delta : |Q| \times \Sigma \times |\Gamma| \to |Q| \times |\Gamma|^{<\ell}$, la máxima cantidad de transiciones de un APD sin ciclar y sin leer de la entrada es menor que

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

(Lemma 2.20 Aho Ullman vol 1). Concluimos la cantidad total de pasos de la derivación para aceptar la cadena de entrada es $3n + 1$ veces esta constante, por lo tanto es lineal en la longitud de la palabra de entrada. □