Lenguajes Formales, Autómatas y Computabilidad

Autómatas de Pila Determinísticos y Gramáticas LR

Segundo Cuatrimestre 2024

Bibliografía para esta clase:
A. V. Aho, J. D. Ullman, The Theory of Parsing, Translation, and Compiling, Vol. 1 y 2, Parsing. Prentice Hall, 1972.
Capítulo 2.6
`https://www-2.dc.uba.ar/staff/becher/Aho-Ullman-Parsing-V1.pdf`
`https://www-2.dc.uba.ar/staff/becher/Aho-Ullman-Parsing-V2.pdf`

John Hopcroft & Jeffrey Ullma, Introduction to Automata Theory, Languages and Computation, Addison Wesley, 1979
`https://www-2.dc.uba.ar/staff/becher/hopcroft.djvu`
Capítulo 10

Autómatas de pila determinísticos

Recordemos

Definición (Autómata de Pila determinístico, página 184 Aho Vol1)

Un autómata de pila $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$, *con*

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$$

es determinístico si para cada $a \in \Sigma$, $q \in Q$ *y* $Z \in \Gamma$ *se cumple que*

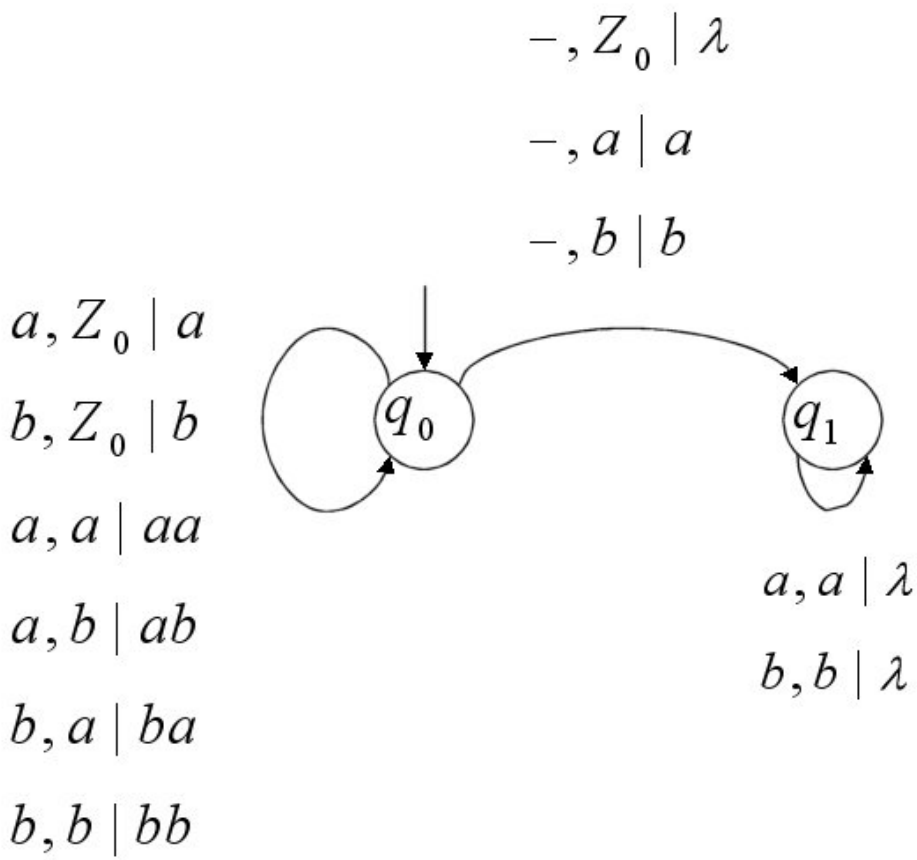
- $\delta(q, a, Z)$ *contiene a lo sumo un elemento y* $\delta(q, \lambda, Z) = \emptyset$

  ó

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

Ejemplo

El siguiente autómata de pila M es determinístico



$$\mathcal{L}(M) = \{\alpha - \alpha^R : \alpha \in (\Sigma \setminus \{'-'\})^*\}.$$

Teorema

No es cierto que para cada autómata de pila no determinístico existe otro determinístico que reconoce el mismo lenguaje.
Es decir, hay lenguajes libres de contexto que no son aceptados por ningún autómata de pila determinístico.

Demostración. $L = \{ww^R\}$ es aceptado por AP no-determinístico pero no es aceptado por ningun AP determinístico ver Hopcroft, Motwani Ulman (2001) página 249:

On the other hand, there are CFL's like L_{wwr} that cannot be $L(P)$ for any DPDA P. A formal proof is complex, but the intuition is transparent. If P is a DPDA accepting L_{wwr}, then given a sequence of 0's, it must store them on the stack, or do something equivalent to count an arbitrary number of 0's. For instance, it could store one X for every two 0's it sees, and use the state to remember whether the number was even or odd.

Suppose P has seen n 0's and then sees 110^n. It must verify that there were n 0's after the 11, and to do so it must pop its stack.[5] Now, P has seen 0^n110^n. If it sees an identical string next, it must accept, because the complete input is of the form ww^R, with $w = 0^n110^n$. However, if it sees 0^m110^m for some $m \neq n$, P must *not* accept. Since its stack is empty, it cannot remember what arbitrary integer n was, and must fail to recognize L_{wwr} correctly. Our conclusion is that:

- The languages accepted by DPDA's by final state properly include the regular languages, but are properly included in the CFL's.

0^n110^n aceptado.
$0^n110^n\ 0^m110^m$, aceptado solamente si $m = n$. Pero la pila ya está vacía!

Lenguaje Determinístico

Recordemos

Definición (Lenguaje libre de contexto determinístico)

Un lenguaje L es libre de contexto determinístico si existe un autómata de pila determinístico (APD) M tal que $L = \mathcal{L}(M)$.

Se reconocen en tiempo lineal.
Son exactamente som lenguajes $LR(k)$ para $k \geq 1$.
Ejemplos
$L = \{a^n b^n\}$
$L = \{a^i b^i a^j : i, j \geq 1\}$

Contraejemplos
$L_0 = \{a^i b^i c^i : i \geq 1\}$ no es libre de contexto (lema de Pumping)

$L_1 = \{a^i b^j a^j : i \neq j\}$ no es libre de contexto (Lema de Ogden)

$L_2 = \{a^i b^j a^j : i \neq j\} \cup \{a^i b^i a^j : i \neq j\}$ no es libre de contexto

$L_3 = \{a^i b^i a^j : i, j \geq 1\} \cup \{a^i b^j a^j : i, j \geq 1\}$ libre de contexto no determinístico

Teorema

El complemento de un lenguaje libre de contexto determinìstico es otro lenguaje libre de contexto determinístico.

Idea ingenua para la demostración, que no funciona.

Dado APD M definir APD M', donde los estados finales pasen a ser no-finales y viceversa. El lenguaje aceptado porM' no necesarimente es el complemento del lenguaje aceptado por M.

1. M puede no consumir totalmente la cadena de entrada

   porque M alcanza una configuración desde la cual ninguna transición es posible.

   ó

   porque M entra en un ciclo infinito de transiciones-λ

   En ambos casos la cadena no consumida es rechazada por ambos autómatas.
2. El autómata M consume toda la cadena de entrada, pero hay alguna cadena que, luego de consumida, deja al autómata en una configuración tal que éste pueda hacer transiciones λ pasando por estados finales y no-finales. El lenguaje aceptado por M' no es el complemento del lenguaje aceptado por M.

Configuraciones ciclantes

Es posible que un autómata de pila determinístico realice una cantidad infinita de λ-movimientos desde alguna configuración, es decir, movimientos que no leen de la cinta de entrada.

Definición (Configuracion ciclante, 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*

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

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.

APD sin ciclos

Teorema (APD sin ciclos- Teorema 2.22 Aho Ullman vol 1, pagina 207)

Para todo APD P hay otro APD P' tal que $L(P) = L(P')$ y P' no tiene configuraciones ciclantes.

Demostración del Teorema APD sin ciclos

Lema (Lema 2.20 de Aho y Ullman vol 1 (pagina 172))

Sea $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ *un APD. Si* $(q, w, A) \overset{n}{\vdash} (q', \lambda, \lambda)$ *entonces para toda* $A \in \Gamma$ *y* $\alpha \in \Gamma^*$ $(q, w, A\alpha) \overset{n}{\vdash} (q', \lambda, \alpha)$.

□

Detección de configuraciones ciclantes APD

Dado APD $P = (Q, \Sigma, \delta, \Gamma, q_0, Z_0, F)$, la función de transición es una función parcial $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to Q \times \Gamma^*$

La máxima cantidad de transiciones que P puede hacer sin leer de la entrada y sin repetir el estado ni el tope de la pila es $|Q| \times |\Gamma|$.

En cada transición se escriben en la pila a lo sumo ℓ símbolos de Γ.

La cantidad máxima de configuraciones distintas sin leer la entrada contando la pila entera es:

$$|Q| \times (\text{la cantidad de posibles pilas de longitud hasta } |Q|\ell|\Gamma|)$$

$$|Q| \sum_{i=0}^{|Q|\ell|\Gamma|} |\Gamma|^i = |Q| \frac{|\Gamma|^{\ell|Q||\Gamma|+1} - 1}{\Gamma - 1}.$$

Este es el máximo de transiciones-λ que P puede hacer sin ciclar.

APD que leen toda la entrada

Definition

Un APD $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ es continuo si para todo $w \in \Sigma^*$ extiste $p \in Q$ y $\alpha \in \Gamma^*$ tales que $(q_0, w, Z_0) \overset{*}{\vdash} (p, \lambda, \alpha)$.

Es decir, APD P es continuo si lee toda la cadena de entrada.

Lema

Para todo APD existe otro equivalente y continuo.

Demostración.

Sea $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ un APD. Construimos APD P' equivalente para que en cualquier configuración donde quede entrada por leer, haya un próximo movimiento.

Algoritmo 2.16 , Aho Ullman vol 1, pag 187

$\mathbf{C_1}$ $= \{(q, Z) |$ (q, λ, Z) es una configuración ciclante y no existe g en F para ningun $\alpha \in \Gamma^*$ tal que $(q, \lambda, Z) \overset{*}{\vdash} (g, \lambda, \alpha)\}$

$\mathbf{C_2}$ $= \{(q, Z) |$ (q, λ, Z) es una configuracion ciclante y hay un $g \in F$ y $\alpha \in \Gamma^*$ tales que $(q, \lambda, Z) \overset{*}{\vdash} (g, \lambda, \alpha)\}$

Supongamos P siempre tiene una próxima transición.
Sea $P' = (Q \cup \{f, t\}, \Sigma, \Gamma, \delta', q_0, Z_0, F \cup \{f\})$ donde f y t son nuevos.
La función $\delta' : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to Q \times \Gamma^*$ se define así:
Para todo $(q, Z) \notin (\mathbf{C_1} \cup \mathbf{C_2})$, $\delta'(q, \lambda, Z) = \delta(q, \lambda, Z)$.
Para todo (q, Z) en $\mathbf{C_1}$, $\delta'(q, \lambda, Z) = (t, Z)$.
Para todo (q, Z) en $\mathbf{C_2}$, $\delta'(q, \lambda, Z) = (f, Z)$.
Para todo $a \in \Sigma$ y $Z \in \Gamma$, $\delta'(f, a, Z) = (t, Z)$ y $\delta'(t, a, Z) = (t, Z)$. □

Demostración del Teorema

Sea APD $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$, continuo.
Definimos $P' = \langle Q', \Sigma, \Gamma, \delta', q_0', Z_0, F' \rangle$ un APD donde

$$Q' = \{[q, k] : q \in Q \text{ y } k = 0, 1, 2\},$$

El propósito de k es indicar si entre transiciones con consumo de entrada en P pasó o no por un estado final.

$$q_0' = \begin{cases} [q_0, 0] & \text{si } q_0 \notin F \\ [q_0, 1] & \text{si } q_0 \in F \end{cases}$$

$$F' = \{[q, 2] : q \in Q\}$$

0 indica que P no paso por F
1 indicaque P sí paso por F
2 indica que P no pasó por F y P va a seguir leyendo.

Para todo $q \in Q$, $[q, 2]$ es estado final al que llega P' antes de que P lea un nuevo simbolo.

La función de transición $\delta' : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to Q \times \Gamma^*$ se define así:

• Si P lee desde q símbolo a, $\delta(q, \lambda, Z) = \emptyset$ y $\delta\,(q, a, Z) = (p, \gamma)$ entonces

$\delta'([q, 0], \lambda, Z) = ([q, 2], Z)$

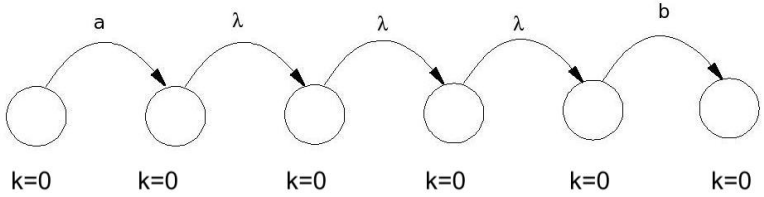
P' acepta la entrada antes de leer a porque P no la aceptó,

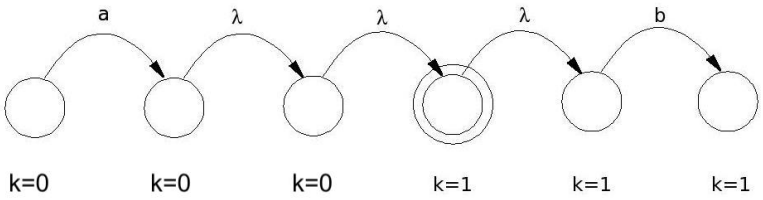
$$\delta'\,([q, 1], a, Z) = \delta'\,([q, 2], a, Z) = \begin{cases} ([p, 0], \gamma), & \text{si } p \notin F \\ ([p, 1], \gamma), & \text{si } p \in F \end{cases}$$

• Si P no lee desde q símbolo a, $\delta\,(q, \lambda, Z) = (p, \gamma)$ y $\delta(q, a, Z) = \emptyset$ entonces

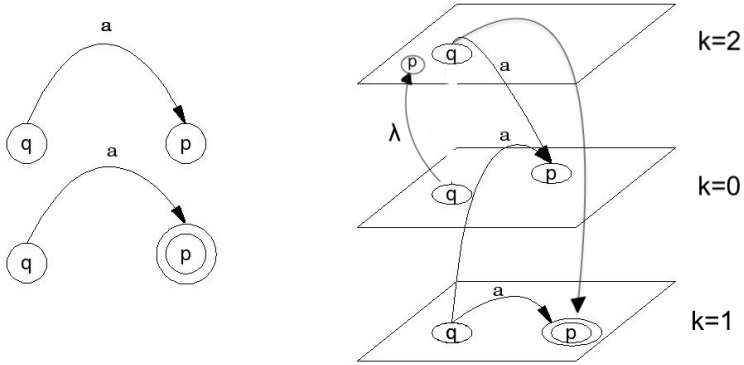
$$\delta'\,([q, 1]\,, \lambda, Z) = ([p, 1], \gamma)$$
$$\delta'\,([q, 0]\,, \lambda, Z) = \begin{cases} ([p, 0], \gamma), & \text{si } p \notin F \\ ([p, 1], \gamma), & \text{si } p \in F \end{cases}$$

Para terminar la demostración debemos ver que para toda $w \in \Sigma^*$, $\delta(q_0, w, Z_0) \in F$ si y solo si $\delta'(q_0', w, Z_0) \notin F'$.

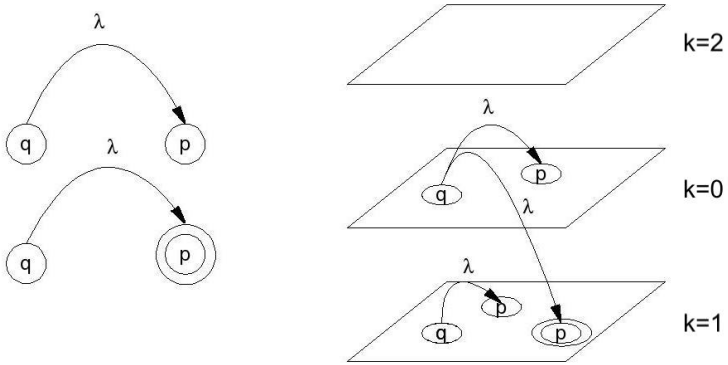

a
λ
λ
λ
b
k=0
k=0
k=0
k=0
k=0
k=0


a
λ
λ
λ
b
k=0
k=0
k=0
k=1
k=1
k=1

M M′


a
q
p
a
q
p
p
q
a
λ
a
p
q
a
q
p
k=2
k=0
k=1

M M′


λ
q
p
λ
q
p
λ
q
p
λ
p
q
λ
λ
p
q
p
k=2
k=0
k=1

Supongamos $w = a_1 \dots a_n \in \Sigma^+$, $\delta(q_0, w, Z_0) \in F$. Llamemos $q_{i_0} = q_0$.

$(q_{i_0}, a_1 \dots a_n, Z_0) \overset{+}{\underset{P}{\vdash}} (q_{i_1}, a_2 \dots a_n, \alpha_1) \overset{+}{\underset{P}{\vdash}} (q_{i_{n-1}}, a_n, \alpha_{n-1} \overset{+}{\underset{P}{\vdash}} (q_{i_n}, \lambda, \alpha_n)$.

Hay dos posibilidades:

O bien $q_{i_n} \in F$

O bien $q_{i_n} \notin F$ y $q_{i_n}, \lambda, \alpha_n) \overset{+}{\underset{P}{\vdash}} (q_{i_m}, \lambda, \alpha_m) \underset{P}{\vdash} (q, \lambda, \alpha), \qquad q \in F$.

Por lo tanto

$([q_{i_0}, j_0], a_1 \dots a_n, Z_0) \overset{+}{\underset{P'}{\vdash}} ([q_{i_1}, j_1], a_2 \dots a_n, \alpha_1) \overset{+}{\underset{P'}{\vdash}}$

$([q_{i_{n-1}}, j_{n-1}], a_n, \alpha_{n-1} \overset{+}{\underset{P'}{\vdash}} ([q_{i_n}, j_n], \lambda, \alpha_n)$.

Si $q_{i_n} \in F$, $j_n = 1$ por lo tanto $\delta'([q_0, j_0], w, Z_0) \notin F'$.

Si $q_{i_n} \notin F$, $([q_{i_n}, j_n], \lambda, \alpha_n) \overset{+}{\vdash} ([q_{i_m}, j_m], \lambda, \alpha_m), \qquad q_m \in F$.
Dado que $q_m \in F$, $i = 1$,por lo tanto $\delta'([q_0, j_0], w, Z_0) \notin F'$.

Para terminar la prueba hay que mostrar que :
si $\delta(q_0, w, Z_0) \notin F$ entonces $\delta'(q_0', w, Z_0) \in F'$.
Para eso veremos que despues de leer toda la entrada P llega a un estado p no final, y eso have que P' llegue a $[p, 0]$ y a $[p, 2]$, que es final. Si P sigue con transiciones λ que lo llevan a q no final, P' irá a estado $[q, 0]$ y $[q, 2]$ que es final.
□

Propiedades de clausura
lenguajes libres de contexto determinísticos

Están clausurados por

Complemento (L) (dimos un automata de pila deterministico)

Interseccion con lenguaje regular (autómata para lenguaje intersección)

Concatenación de L deterministico seguido de lenguaje regular (automata).

$pre(L)$: subconjunto de cadenas de L con un prefijo propio en L

$Min(L)$ cadenas de L no tienen un prefijo propio en L.

$Max(L)$ cadenas de L que no son un prefijo de una cadena más larga en L

Propiedades de clausura de lenguajes deterministicos

No están clausurados por

interseccion
union
reversa
concatenacion
clausura de Kleene

Lenguajes determinísticos no están clausurados por Intersección

$L_1 = \{a^i b^i c^j : i, j \geq 0\}$ es LC deterministico
$L_2 = \{a^i b^j c^j : i, j \geq 0\}$ es LC deterministico
$L_1 \cap L_2 = \{a^n b^n c^n : n \geq 0\}$ no es libre de contexto (Pumping)

Lenguajes determinísticos no están clausurados por union

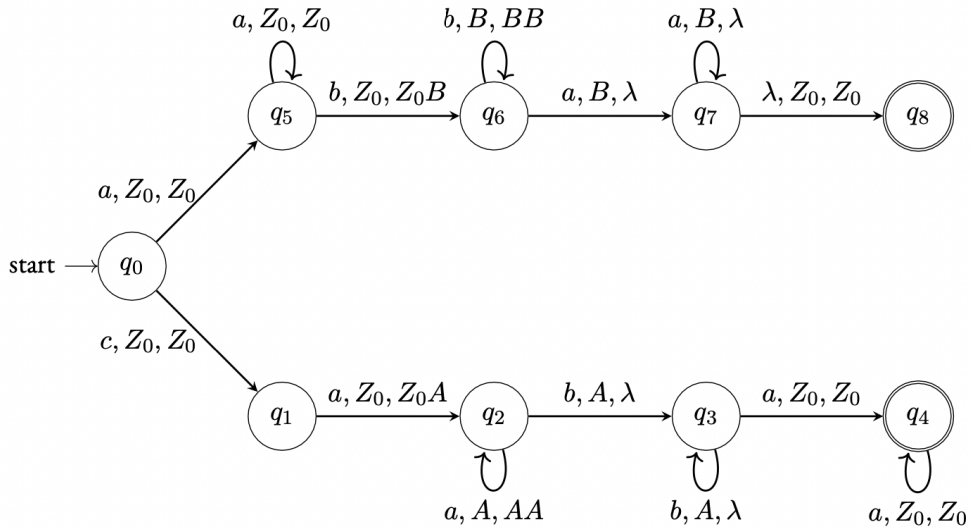
Sabemos que están clasurados por complemento.
Si estuvieran clasurados por unión, también lo estarían por intersección porque

$$L_1 \cap L_2 = \overline{\overline{L_1} \cap \overline{L_1}}$$

Y ya demostramos que no estan clasurados por intersección.

Lenguajes determinísticos no están cerrados por reversa

$L = \{ca^i b^i a^j : i, j \geq 1\} \cup \{c^2 a^i b^j a^j : i, j \geq 1\}$ es LC deterministico



Supongamos
$L^R = \{a^j b^j a^i c^2 : i, j \geq 1\} \cup \{a^j b^i a^i c : i, j \geq 1\}$ es LC deterministico.
Quitar las c's del final debería dejar un lenguaje determinístico.
Sin embargo,
$L^R = \{a^j b^j a^i : i, j \geq 1\} \cup \{a^j b^i a^i : i, j \geq 1\}$ es LC no deterministico.

Lenguajes determinísticos no están cerrados por concatenación

Sea $R = \{c, c^2\}$.

Sea $L = \{ca^i b^i a^j : i, j \geq 1\} \cup \{a^i b^j a^j : i, j \geq 1\}$ (es LC determinístico)

$RL \cap c^2 a^* b^* c^* = c^2 L'$, con $L' = \{a^i b^i a^j : i, j \geq 1\} \cup \{a^i b^j a^j : i, j \geq 1\}$.

debería ser LC-determinístico,

porque interseccion de LC-determinístico y regular es LC-determinístico.
Y tambien L' deberia ser LC-determinístico porque quitar dos transiciones iniciales no afectaría.

Sin embargo, sabemos que L' es LC-no-determinístico.

¡Atención!
Si L es LC-deterministico y R es regular entonces LR es LC-deterministico.

Lenguajes determinísticos no están cerrados por clausura Kleene

Sea $L = \{a^i b^i a^j : i, j \geq 1\} \cup \{ca^i b^j a^j : i, j \geq 1\} \cup \{c\}$, que es LC-determinístico.
Entonces
$L^* \cap c^2 a^+ b^+ a^+ = c^2(\{a^i b^i a^j : i, j \geq 1\} \cup \{a^i b^j a^j : i, j \geq 1\})$.
Si L^* fuera LC-deterministico tambien lo sería
$\{a^i b^i a^j : i, j \geq 1\} \cup \{a^i b^j a^j : i, j \geq 1\}$.
Pero no lo es.

Decisión sobre lenguajes determinísticos

Sea L determinístico.

Hay algoritmos para :

$L = \emptyset$? (igual que caso no determinístico)
L finito? (pumping)
L infinito? (pumping)
L cofinito? (pumping sobre L complemento)
$L = \Sigma^*$? ($\overline{L} = \emptyset$?)
$L_1 = L_2$? Géraud Sénizergues, 2002 Gödel Prize , for proving in 1997 that equivalence of deterministic pushdown automata is decidable
$w \in L$? en tiempo lineal en la longitud de w (algoritmo LR).
L es regular? R. Steans. A Regularity Test for Pushdown Machines. Information and Control 11, 323-340,1967
$L = R$?
$L \subseteq R$?

No hay algoritmos para:

$L_1 \subseteq L_2$?
$L_1 \cap L_2 = \emptyset$?

Libre de contexto Determinístico implica No Ambiguo

Recordemos

Definición

Para todo $A \in V_N$, $\alpha_1, \alpha_2 \in (V_T \cup V_N)^*$,

- $\alpha_1 A \alpha_2 \underset{L}{\Rightarrow} \alpha_1 \alpha' \alpha_2$ *es una drivación más a la izquierda si* $A \to \alpha' \in P$ *y* $\alpha_1 \in V_T^*$.
- $\alpha_1 A \alpha_2 \underset{R}{\Rightarrow} \alpha_1 \alpha' \alpha_2$ *es una derivación más a la derecha si* $A \to \alpha' \in P$ *y* $\alpha_2 \in V_T^*$.

Definición (Gramáticas ambiguas)

Una gramática libre de contexto G *es ambigua si existe* $\alpha \in \mathcal{L}(G)$ *con más de una derivación más a la izquierda.*

Los lenguajes determinísticos libres de contexto admiten una gramática libre de contexto no ambigua.

Y hay una clase importante de lenguajes no determinísticos libres de contexto que también admiten una gramática libre de contexto no ambigua.

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 $j > 0$, $\ell > 0$,

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

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 $LR(k)$.

Autómata de Pila Enriquecidos

Autómatas con un contador.
Son autómatas donde el alfabeto de pila tiene exactamente dos símbolos: Z_0 y I. en cada transición pueden revisr si el contador es Z_0, o no, y pueden incrementar o decrementarlo.

Los lenguajes reconocidos por auómatas contadores incluyen a todos los lenguajes regulares pero son un subconjunto propio de los lenguajes reconocidos por autótas de pila.

Autómata de Pila Enriquecidos

Los autómatas de pila de dos vías, tienen una pila, control finito y una cinta cuya cabeza puede moverse en ambas direcciones (Aho, Hopcroft, Ullman 1968, and Gray, Harrison, Ibarra 1967). Estos autómatas tiene menos poder computacional que una máquina de Turing.

Los autómatas con dos o más pilas son equivalentes a una máquina de Turing (Aho Ullman Teoremas 8.13, 8.14 y 8.15).

Los autómatas con tres contadores son equivalentes a una máquina de Turing (Aho Ullman Teoremas 8.13, 8.14 y 8.15).

Los autómatas con dos contadores son equivalentes a una máquina de Turing (Aho Ullman Teoremas 8.13, 8.14 y 8.15).

Los autómatas con una cola son equivalentes a autómatas con sos pilas, por lo tanto, equivalentes a una máquina de Turing (Bisbay 1959).