Teoría de Lenguajes
Clase 10: Autómatas de Pila Determinísticos

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`
Capítulo 2.6

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.*

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*

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

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.

APD Sin ciclos!

Teorema (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.

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

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 del Lema: un APD continuo equivalente

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

$$
\begin{aligned}
\mathbf{C_1} &= \{(q,Z) \mid && (q,\lambda,Z) \text{ es una configuración ciclante y no existe } g \text{ en } F \\
& && \text{para ningun } \alpha \in \Gamma^* \text{ tal que } (q,\lambda,Z) \overset{*}{\vdash} (g,\lambda,\alpha)\} \\
\mathbf{C_2} &= \{(q,Z) \mid && (q,\lambda,Z) \text{ es una configuracion ciclante y hay un } g \in F \text{ y} \\
& && \alpha \in \Gamma^* \text{ tales que } (q,\lambda,Z) \overset{*}{\vdash} (g,\lambda,\alpha)\}
\end{aligned}
$$

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 $\delta'(t,a,Z) = (t,Z)$. $\square$

Demostración del Teorema

Dado el APD $P = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$, que por lo ya visto podemos considerar que consume toda la cadena de entrada, 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 detectar si entre transiciones con consumo de entrada pasó o no por un estado final.
Si pasó por un estado final, entonces se hará $k = 1$, sino se hará $k = 2$.

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

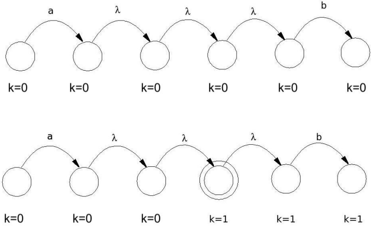
$$q_0' = \begin{cases} [q_0, 0] & \text{si} \ \ q_0 \notin F \\ [q_0, 1] & \text{si} \ \ q_0 \in F \end{cases}$$

Los estados $[q, 0]$ dicen que P no pasó por estado final desde su última λ-transición.

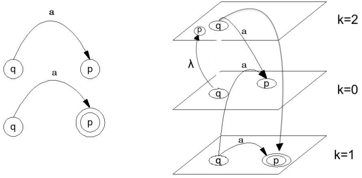
Los estados $[q, 1]$ dicen que P sí pasó por estado final desde su última λ-transición.

Los estados $[q, 2]$ son finales.

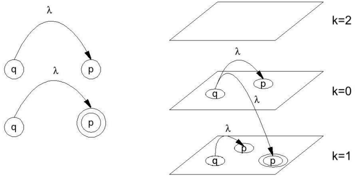
Si P' está en un $[q, 0]$ y P está por hacer una transición que sí lee de la entrada, entonces P' primero entra en estado $[q, 2]$ y luego simula a P. Luego P' aceptará exactamente cuando P no acepte. El hecho de que P es continuo asegura que P' siempre consigue aceptar una entrada cuando P no la acepta.


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
k=2
p
q
a
λ
a
p
k=0
q
a
k=1
q
p

M M’


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

La función de transición δ' se define así:

• Si $\delta\left(q,a,Z\right)=\left(p,\gamma\right)$ entonces

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

Dado que $\delta\left(q,a,Z\right)=\left(p,\gamma\right)$ y P es determinístico tenemos $\delta(q,\lambda,Z)=\emptyset$, por lo tanto definimos

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

Esto hace que P' acepte una entrada exactamente cuando P no la acepta.

• Si $\delta\left(q,\lambda,Z\right)=\left(p,\gamma\right)$ entonces

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

Propiedades de clausura
lenguajes libres de contexto determinísticos

Están clausurados por

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

$pre(L)$: 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

Interseccion con lenguaje regular (autómata para lenguaje intersección)

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

Propiedades de clausura
lenguajes libres de contexto deterministicos

No están clausurados por

intersección
union
reversa
concatenacion
clausura de Kleene

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 siquiera libre de contexto (Pumping)

No están clausurados por union

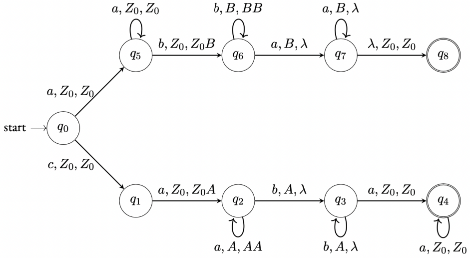
Sabemos que están clasurados por complemento.
Si estuvieran clasurados por unión, también lo estarían por intersección porque

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

Y ya demostramos que no estan clasurados por intersección.

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.

No están cerrados por concatenación

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

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

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

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.

No están cerrados por clausura Kleene

Sea $L = \{a^i b^i a^j : i,j \geq 1\} \cup \{c a^i b^j a^j : i,j \geq 1\} \cup \{c\}$, 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 libres de contexto determinísticos

Sea L LC-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.

Apéndice : Lema de Ogden

Sea L lenguaje generado por gramática libre de contexto. Existe $p \geq 1$ tal que para todo $\forall w \in L$ de longitud mayor o igual que p y cualquier modo de marcar al menos p posiciones hay un no terminal A y una forma de escribir $w = uxzyv$, tal que
$S \Rightarrow^* uAv \Rightarrow^* uxAyv \Rightarrow^* uxzyv$
z contienen al menos una posicion marcadas
xzy contiene al menos p posiciones marcadas
u, x ambas contienen posiciones marcadas,
ó
y, v ambas contienen posiciones marcadas.